

UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Information Technology Curriculum

Eduard Sing
**A Prototype to Transform Models of Secure Tropos
and Misuse Case Diagrams**
Bachelor's Thesis (6 ECTS)

Supervisor: Dr. Raimundas Matulevičius

Tartu 2014

A Prototype to Transform Models of Secure Tropos and Misuse Case Diagrams

Abstract

Main product of the the Digital Age is information and main tool of this Age is an information system (IS). The purpose of IS is accumulating, storing, multiplying and distributing information. To be sure that data and information will be stored and distributed as secure as possible, it is necessary to pay attention to the security design of an IS from the very early stages of the software developing process. Security modeling languages such as Misuse cases and Secure Tropos may help to deal with this problem. These two languages have proven their usefulness to elicit, negotiate and visualize security requirements and contribute to the thorough definition of the secure information systems. Although these languages are serving for identical aim, they differ in many ways. Main difference are various viewpoints on the modeled system, *i.e.*, Secure Tropos help to understand the security rationale, Misuse cases help to relate security and functionality together.

When designing secure software it is important to consider different viewpoints to the problem. But the problem of using two or more languages is to translate different viewpoints consistently.

This problem can be solved with automation of the transformation process between two modeling languages. Despite Misuse cases and Secure Tropos have differences, they also have similarities, allowing us to automate transformation process from one language to another and *vice versa*.

The main goal of this thesis is to develop a prototype that would facility and support transformation of the Misuse cases to Secure Tropos models and *vice versa*. The prototype would allow developers to interchange the data between two modeling softwares, where models of two languages can be modeled using MagicDraw UML (Misuse cases) and SecTro2 Tool (Secure Tropos).

To ensure that automated transformation process is efficient than manual transformation process we have conducted some practical experiment. It showed that prototype allows developers to shorten the time that is spent on transformation. The survey showed that prototype is user friendly.

Keywords

Secure Tropos models, Misuse case diagrams, transformation, transformation rules, security modeling languages, information system security, MagicDraw UML, SecTro2, plug-in, prototype

Secure Tropos mudelite ja Misuse Case diagrammide teisendamise prototüüp.

Lühikokkuvõtte

Peamiseks tooteks 21. sajandil on informatsioon, ja peamiseks vahendiks informatsiooni vahetamises, säilitamises ja kogumises on infosüsteemid. Et kindlustada informatsiooni turvalisust hoidmisel ja levitamisel on väga oluline pöörata oma tähelepanu infosüsteemide turvalisusele juba selle arendamisprotsessi algstaadiumitel. Sellel juhul kasutavad arendajad tihti modelleerimiskeeli, nagu *Secure Tropos* ja *Misuse cases*. Need kaks tõestasid oma võimet tuua esile, visualiseerida ja analüüsida infosüsteemide turvanõudeid ja turvariske, panustades infosüsteemide turvalisusele. Kuigi need modelleerimiskeeled teenivad identsete eesmärkide nimel, erinevad nad mitmeti. Peamiseks erinevuseks on erinevate vaatenurkade olemasolu – nt. *Secure Tropos* mudelid aitavad mõista turvalisuse loogikat ja *Misuse case* diagrammid seovad turvalisust ja funktsionaalsust kokku.

Infosüsteemi turvalisuse projekteerimisel on väga oluline kinni pidada paljudest eesmärkidest, vaadates probleemile kõikvõimalikest vaatenurkadest ja kasutades modelleerimisel kohe mitut modelleerimiskeelt. Selline lähenemine aitab infosüsteemi turvalisust tunduvalt tõsta, kuid nõuab mudelitest järjepidevat uuendamist. Sellel juhul peamiseks probleemiks on teisendamise järjekindluse säilitamine.

Selle probleemi lahenduseks võiks olla tööriist, mis saaks automatiseerida teisendamise protsessi ühest modelleerimise keelest teise ja vastupidi. Vaatamata *Misuse case* ja *Secure Tropos* erinevustele, nendel kahel on ka sarnasusi, mis võimaldavad defineerida teisendamise reegleid arvuti abil ja viia ellu teisendamise prototüübi.

Selle lõputöö eesmärgiks on arendada tööriista prototüüp, mis saaks teisendada *Secure Tropos* mudeli *Misuse case* diagrammiks ning vastupidi. Prototüübi abiga saavad arendajad jagada andmeid kahe modelleerimistööriista vahel, kus *Secure Tropos* mudelid (SecTro2 Tool) ja *Misuse case* diagrammid (MagicDraw UML) on projekteeritud.

Et osutada automatiseeritud protsessi paremust mitteautomatiseeritud protsessi üle, me viime läbi küsitluse ja väikese praktilise katse. Praktiline katse saab tõestada, et teisendamine prototüübiga on oluliselt kiirem. Küsitlus aitab näidata, et prototüüp on piisavalt kasutajasõbralik tavalise kasutaja jaoks.

Võtmesõnad

Secure Tropos mudelid, Misuse case diagrammid, teisendamine, teisendamise reeglid, turvalisuse modelleerimis keeled, infosüsteemide turve, MagicDraw UML, SecTro2, lisamoodul, prototüüp

Contents

Abbreviations and Acronyms.....	7
Chapter 1: Introduction.....	8
Chapter 2: ISSRM Domain Model.....	9
2.1. Asset-related concepts.....	9
2.2. Risk-related concepts.....	9
2.3. Risk treatment-related concepts.....	10
2.4. ISSRM Process.....	10
2.5. Summary.....	11
Chapter 3: Security Modeling Languages.....	12
3.1. Security Modeling Language.....	12
3.1. Misuse Cases.....	12
3.1.1. Misuse Cases in Terms of ISSRM.....	12
3.1.2. MagicDraw UML.....	13
3.2. Secure Tropos.....	14
3.2.1. Secure Tropos in Terms of ISSRM.....	14
3.2.2. SecTro2.....	14
3.3. Summary.....	15
Chapter 4: Transformation.....	16
4.1. Software Limitations.....	16
4.2. Transformation Rule Types.....	16
4.3. Secure Tropos to Misuse Cases Transformation.....	16
4.3.1. Model Example.....	16
4.3.2. Transforming Asset-related Concepts.....	18
4.3.3. Transforming Risk-related Concepts.....	18
4.3.4. Transforming Risk Treatment-related Concepts.....	19
4.3.5. Finalizing Transformation.....	20
4.4. Misuse Cases to Secure Tropos Transformation.....	20
4.4.1. Diagram Example.....	20
4.4.2. Transforming Asset-related Concepts.....	21
4.4.3. Transforming Risk-related Concepts.....	22
4.4.4. Transforming Risk Treatment-related Concepts.....	23
4.4.5. Finalizing Transformation.....	23
4.5. Summary.....	24
Chapter 5: Prototype.....	25
5.1. Design and Requirements.....	25
5.1.1. Requirements for Misuse Case to Secure Tropos Transformation.....	25
5.1.2. Requirements for Secure Tropos to Misuse Case Transformation.....	26
5.2. Implementation.....	27
5.2.1. Transformation of Misuse Cases to Secure Tropos.....	28
5.2.2. Transformation of Secure Tropos to Misuse Cases.....	28
5.3. Summary.....	31
Chapter 6: Validation.....	32
6.1. Scope.....	32
6.2. Validation Process.....	32
6.3. Results.....	32
6.4. Threats to Validity.....	34
6.5. Summary.....	34
Chapter 7: Conclusion.....	35
7.1. Limitations.....	35
7.2. Conclusions.....	35
7.3. Future Work.....	35

Bibliography.....	36
Appendices.....	38
Appendix A: Alignment Tables.....	38
Appendix B: Transformation Rules Examples.....	44
Appendix C: Validation Questionnaire.....	54
Appendix D: Models for Validation.....	55

Index of Tables

Table 1: Misuse case elements stereotype table.....	28
Table 2: Calculated mean time (in minutes) for transformation processes.....	33
Table 3: Interview overall results (ST -Secure Tropos; MUC – Misuse cases).....	34
Table 4: SROMUC asset-related constructs in terms of ISSRM (adapted from [Soomro and Ahmed, 2013]).....	38
Table 5: SROMUC risk-related constructs in terms of ISSRM (adapted from [Soomro and Ahmed, 2013]).....	39
Table 6: SROMUC risk treatment-related constructs in terms of ISSRM (adapted from [Soomro and Ahmed, 2013]).....	40
Table 7: Secure Tropos constructs in terms of asset-related concepts of ISSRM (adapted from [Matulevičius et al., 2012]).....	41
Table 8: Secure Tropos risk-related constructs in terms of ISSRM (adapted from [Matulevičius et al., 2012]).....	42
Table 9: Secure Tropos risk-treatment constructs in terms of ISSRM (adapted from [Matulevičius et al., 2012]).....	43
Table 10: Examples of Secure Tropos constructs transformed to Misuse case constructs.....	44
Table 11: Examples of Misuse case constructs transformed to Secure Tropos constructs.....	49

Index of Figures

Figure 1: The ISSRM domain model (adapted from [Dubois et al., 2010]).....	10
Figure 2: ISSRM process (adapted from [Mayer, 2009]).....	11
Figure 3: MagicDraw UML user interface.....	13
Figure 4: SecTro2 tool user interface.....	15
Figure 5: eSap model example modeled with SecTro2 tool, Security Requirements view.....	17
Figure 6: eSap model example "Authentication attack" threat sub-view.....	17
Figure 7: Misuse case result diagram (STMC10 and STMC11 rule example).....	20
Figure 8: Secure Tropos result model (Threat “Threat-1001” sub-view) (MCST9 rule example).....	22
Figure 9: Secure Tropos result model (Security Requirements view) (MCST11 and MCST12 rule example).....	23
Figure 10: Main concept of the prototype.....	25
Figure 11: Use cases for Misuse case to Secure Tropos transformation process functional requirements.....	26
Figure 12: Use cases for Secure Tropos to Misuse case transformation process functional requirements.....	27
Figure 13: Plug-in package structure.....	29
Figure 14: plugin package Java classes.....	29
Figure 15: plugin.model package Java classes.....	30
Figure 16: plugin.transformation package Java classes.....	30

Figure 17: STMC1 rule example.....	44
Figure 18: STMC2 rule example.....	45
Figure 19: STMC3 rule example.....	45
Figure 20: STMC4 rule example.....	46
Figure 21: STMC5 rule example.....	46
Figure 22: STMC6 rule example.....	47
Figure 23: STMC7 rule example.....	47
Figure 24: STMC8 rule example.....	48
Figure 25: STMC9 rule example.....	48
Figure 26: MCST1 rule example (Security Requirements view).....	49
Figure 27: MCST2 rule example (Security Requirements view).....	50
Figure 28: MCST3 rule example (Security Requirements view).....	50
Figure 29: MCST4 rule example (Security Requirements view).....	51
Figure 30: MCST5 rule example (Security Requirements view).....	51
Figure 31: MCST6 rule example (Security Requirements view).....	52
Figure 32: MCST7 rule example (Threat "Threat-1001" sub-view).....	52
Figure 33: MCST8 rule example (Threat "Threat-1001" sub-view).....	53
Figure 34: MCST10 rule example (Security Requirements view).....	53
Figure 35: „Meeting scheduler“ Secure Tropos model example for validation.....	55
Figure 36: "Disclose Agreement" Threat sub-view.....	55
Figure 37: Misuse case diagram "Meeting scheduler" example for validation.....	56

Abbreviations and Acronyms

IS	Information System
ISSRM	Information System Security Risk Management
MUC	Misuse cases
OpenAPI	Open Application Program Interface
SROMUC	Security risk-oriented Misuse cases
ST	Secure Tropos
UML	Unified Modeling Language
XML	Extensible Markup Language

Chapter 1: Introduction

The security modeling languages such as Secure Tropos [Mouratidis and Giorgini, 2007] and Misuse cases [Sindre and Opdahl, 2005] can contribute to the security of IS from the early stages of system development. Secure Tropos and Misuse cases are different and have opposite viewpoints on the security definition. The Secure Tropos helps to define the security mechanisms and methods that are contributing to satisfaction of the security constraints. Misuse cases help to define security requirements of the IS. The usage of these two together helps to express various security requirements and security mechanisms, allowing developers to improve the overall security of the IS. The main problem of using these two modeling languages simultaneously is the difficulty to develop and maintain the consistency of the system model.

Although Secure Tropos and Misuse cases have differences, they also have similarities. Both languages are aligned to the domain model of Information System Security Risk Management (ISSRM) [Mayer, 2009]. The alignment clearly defines how modeling language constructs can represent the constructs of ISSRM. The alignment of Secure Tropos and Misuse cases helps to spot the similar patterns of both modeling languages, allowing to define the transformation rules between models of Secure Tropos and Misuse case diagrams. The automation of these transformation rules could solve the time consumption problem. This work raises two questions: (i) How to manage security risks using different modeling languages and to keep model consistency? and (ii) How to automate model transformation between Secure Tropos and Misuse cases?

This thesis main goal is to develop a prototype that would facilitate and support automated transformation of the Misuse cases to Secure Tropos models and *vice versa*. The prototype would allow developers to interchange the data between two modeling softwares, where models of two languages can be modeled – MagicDraw UML (Misuse cases) and SecTro2 Tool (Secure Tropos). The main base of the prototype would be defined using the OpenAPI of industrial tool MagicDraw UML, and would be acting as a plug-in tool. To validate the prototype we will conduct the validation process.

The thesis is divided to seven chapters and is structured as follows. First chapter introduces thesis overall purposes and main goals; explains thesis structure. Second, third and fourth chapters are introducing the overall background of the work. Second chapter explains the ISSRM domain model, its concepts and process. Third chapter introduces two modeling languages – Secure Tropos and Misuse cases, discusses the language extensions, shows the alignment of modeling languages with ISSRM and announces the software tools for modeling. Fourth chapter discusses and explains the transformation rules between two languages. The fifth and sixth chapters contribute to explanation of the developed prototype. Fifth chapter gives details about implementation of the prototype, discusses it's design, code structure and requirements. Chapter 6 introduces the validation process of the prototype. The seventh chapter concludes the overall work.

This work also includes appendices: A, B, C, D are located at the end of the thesis; E, F, G, H are located on the additional CD. Appendices A, B, C, D contain additional information related to thesis: alignment tables, transformation rule examples, validation questionnaire and examples of models used in validation. Appendices E, F, G respectively contain the documentation for source code of the plug-in tool, plug-in source code, compiled plug-in tool and report generation templates.

Chapter 2: ISSRM Domain Model

Information System Security Risk Management (ISSRM) domain model is set of security methods, standards and frameworks, which combines several risk management concepts and methodologies such as Risk Management and Security Risk Management. ISSRM domain model helps to define, and handle security risks and threats within the design of the software system during security engineering process. ISSRM is made specially for IS developing process [Mayer, 2009].

Concepts of the ISSRM domain model can be divided into three different categories as shown below in the Figure 1: (i) asset-related (yellow), (ii) risk-related (orange), and (iii) risk treatment-related concepts (green). Main six steps (see Figure 2) of ISSRM process are: (i) Context and Asset identification, (ii) Determination of Security Objectives, (iii) Risk Analysis and Assessment, (iv) Risk Treatment, (v) Security Requirements Definition and (vi) Control Selection and Implementation [Mayer, 2009][Soomro, 2012].

2.1. Asset-related concepts

Asset-related concepts represent the instances that have to be protected by security mechanisms, in other words – assets are all valuable and essential entities of organization that have to be protected [Dubois *et al.*, 2010]:

- *Asset* – the object or resource that is used by the company in order to achieve certain goals in their business activities (*e.g.*, technology, currency). Can be structured to *IS-assets* and *business assets*;
- *Business asset* – the process or resource of the company that can perform informational manipulations in order to achieve business goals of organization (*e.g.*, information technology, employee skills);
- *IS-asset* – the supporting object, component, property or resource of IS, that is used by business asset to process information in certain way (*e.g.*, database, computer);
- *Security criterion* – criteria or property of business asset, that is defined to constraint business assets in order to achieve security goals of the IS, such as integrity or confidentiality;

2.2. Risk-related concepts

Risk-related concepts represent the instances that potentially can damage, threat or harm the assets of IS [Dubois *et al.*, 2010]:

- *Risk* – is a combination of one *event* and one or more *impacts*; represents the theoretical possibility of harming and damaging *assets*;
- *Impact* – the negative consequences of the *risk*, predicts what damage can be done to *assets* of the IS (*e.g.*, data loss, loss of confidentiality);
- *Threat* – a plan or intention of *threat agent* to inflict damage to *assets*;
- *Vulnerability* – the flaw, weakness or shortage of *asset*, can be used by attacker to produce *threat*. *Vulnerability* can be attacked, exploited or used by *threat agent* with intentions to harm *asset*;
- *Event* – set of actions, that consists of the *threat* and *vulnerability* or *vulnerabilities*. *Event* leads to *impact*;
- *Threat agent* – person with intentions to damage the *assets* of IS. Uses *attack methods* to exploit *vulnerabilities* of the IS. *Threat agent* is producing *threats*;
- *Attack method* – method that can be used by *threat agent* in order to achieve agent's goals, exploiting *vulnerabilities*;

2.3. Risk treatment-related concepts

Risk treatment-related concepts represent the actions and methods that help to protect assets from being harmed and damaged by risk-related concepts [Dubois *et al.*, 2010]:

- *Risk treatment* – manipulations or countermeasures, that contribute in reducing or avoiding the *risk*;
- *Security requirement* – the proposed improvement to mitigate one ore many *risks*;
- *Control* – the active countermeasure to prevent *threats* (e.g., firewall);

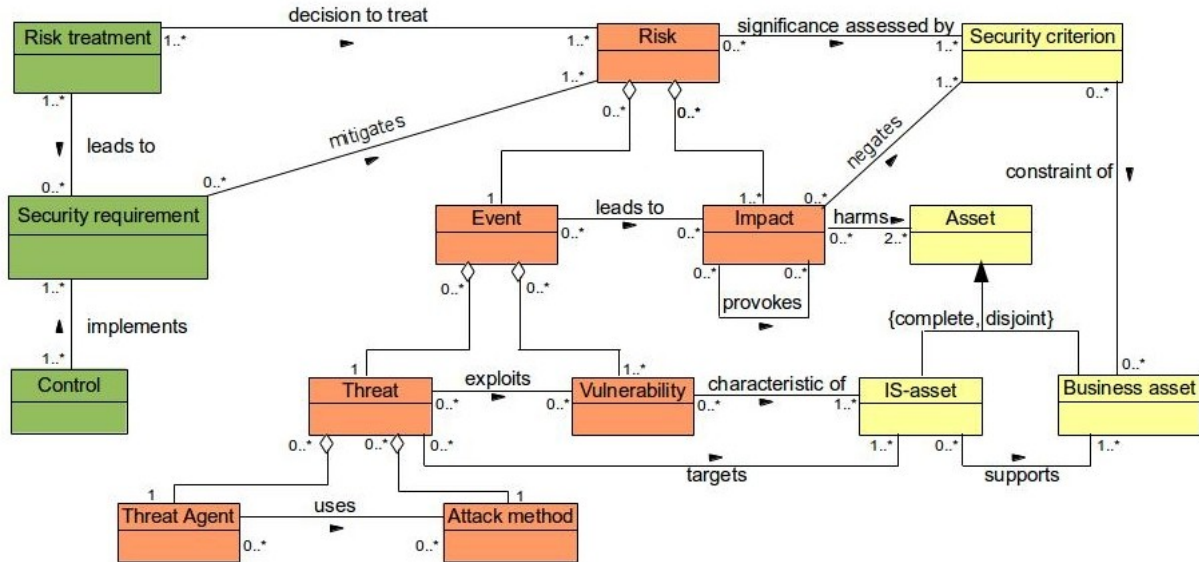


Figure 1: The ISSRM domain model (adapted from [Dubois *et al.*, 2010])

2.4. ISSRM Process

ISSRM process is a model based engineering process, which consists of the six steps. The process is iterative and it may be repeated until reaching the satisfying level of security (see Figure 2) [Matulevičius *et al.*, 2012][Mayer, 2009]:

1. **Context and Asset Identification:** The main goal of the first step is to define organization context and identify *assets* of the company, focusing on sensitive activities related to informational security. First of all are defined *business assets* and afterwards can be identified *IS assets*;
2. **Determination of Security Objectives:** Relying on the previous step, organization determines it's security objectives and properties of the *assets*. Usually security objectives can be described as integrity, confidentiality and availability.
3. **Risk Analysis and Assessment:** During this step organization identifies risks and performs the risk estimation analysis. If found risks are evaluated against estimated security objectives then process can proceed to the next step, otherwise process needs to start over from the first step;
4. **Risk Treatment:** This step can be performed using four different strategies: (i) risk avoidance – no decision is made, avoiding involvement in any risk situation; (ii) risk reduction – perform actions in order to reduce the probability of negative consequences made by risk impact; (iii) risk retention – the negative consequences of the risk impact are not serious, hence no actions to mitigate risk are taken; (iv) risk transfer – the actions reducing the risk probability are shared between several mechanisms or passed to other party;

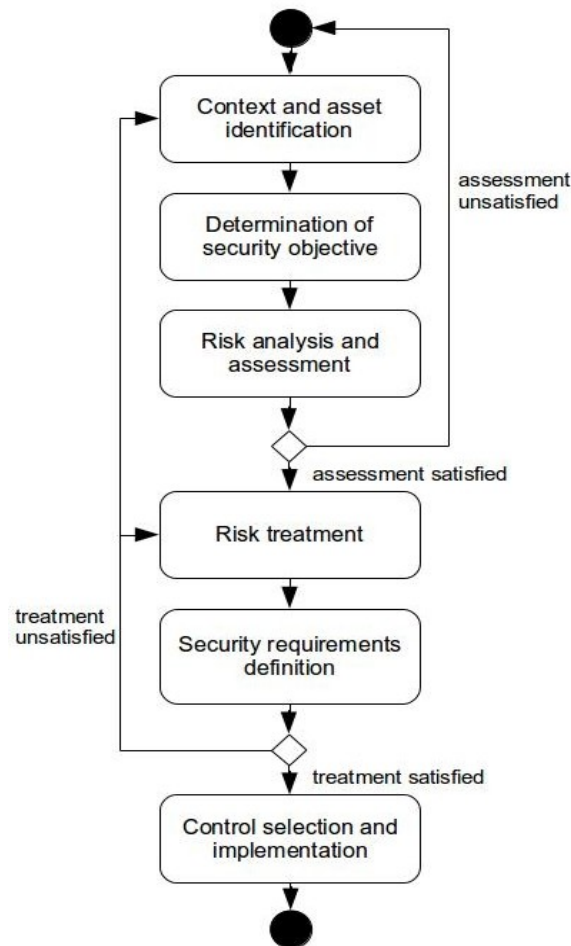


Figure 2: ISSRM process (adapted from [Mayer, 2009])

5. **Security Requirements Definition:** The actions in this step depend on previously chosen risk treatment strategy. Risk reduction assumes defining security requirements that will mitigate security risks. Security requirements for risk transformation strategy may be fulfilled by third party. After security requirements are defined, they need to be verified against security they provide. If proposed security requirements are satisfying security needs - process can proceed to the next step; otherwise either risk treatment step needs to be repeated, either all previous steps from the beginning;
6. **Control Selection and Implementation:** In this step security of the system is improved by implementation, countermeasures and definition of the appropriate security policies;

2.5. Summary

ISSRM domain model is a set of security methods, standards and frameworks. ISSRM domain model helps to define security risks, security requirements and security mechanisms during IS security engineering process. The concepts of ISSRM domain model are divided in three categories: (i) asset-related concepts, (ii) risk-related concepts and (iii) risk-treatment concepts. ISSRM process is a model based engineering process, which consists of the six steps.

Chapter 3: Security Modeling Languages

This chapter introduces two security modeling languages – Secure Tropos and Misuse cases (MUC); discusses their theoretical background, modeling rules, software tools and provides examples of designed models and diagrams.

3.1. Security Modeling Language

The modeling language is artificial language applied in requirement modeling stages of the software developing process. As the ordinary languages, modeling languages have their own syntax, rules, meanings and semantics. Main purposes of modeling language are: (i) specify and show the future behavior of the software under development; (ii) define and represent the abstract and concrete objects or events related to software and its behavioral processes. The activity of using modeling language is called modeling.

The security modeling language is particularistic type of the modeling language, which aims on the understanding, analyzing, modeling and handling the security risks of the software system under development. Security modeling languages (*e.g.*, Secure Tropos, Misuse cases, KAOS extension to Security [Mayer, 2009], BPMN [Silver, 2009], Mal-activities [Sindre, 2007], Secure UML [Lodderstedt *et al.*, 2002]) are mainly used during the security engineering stage of the software development process, helping to model security risks, security mechanisms and security requirements of software system. To meet security standards security modeling languages are aligned with particular Risk Management and Security Risk Management methodologies (*e.g.*, ISSRM).

3.1. Misuse Cases

Misuse cases (MUC) is a security modeling language, extended from Use case models by [Sindre and Opdahl, 2005]. Authors of the MUC found that use cases by it selves are very suitable at the defining functional requirements, but do not allow developers to specify security-related problems, moreover developers had to use different modeling languages to explain security requirements; therefore were proposed additions to Use case models [Sindre and Opdahl, 2005].

The main idea of Misuse case models is the opposite concept of Use cases, where the unwanted activity can be defined using negative entities of an *actor* and *use case* – *misuser* and *misuse case*. To allow developers to specify countermeasures against the unwanted activity were proposed additional entities: *security use case*, and additional relationships *mitigates* and *threatens* [Sindre and Opdahl, 2005].

The further studies in the same field [Matulevičius *et al.*, 2012][Soomro and Ahmed, 2013] have shown that MUC is very useful at modeling security risks, but the countermeasure modeling step has a serious lack in the constructs, leading to “misinterpretation of the security-related concepts” and “poor security solutions”. According to these studies were proposed several additional constructs, transforming MUC to extended MUC – Security risk-oriented Misuse cases (SROMUC); also was made the alignment of SROMUC to ISSRM [Soomro and Ahmed, 2013].

3.1.1. Misuse Cases in Terms of ISSRM

The alignment of SROMUC was made by [Soomro and Ahmed, 2013]. In this part we show how SROMUC constructs are aligned with ISSRM concepts in three stages: (i) asset-related concepts; (ii) risk-related concepts; (iii) risk treatment-related concepts.

Asset-related concepts (see Appendix A: Alignment Tables, Table 4) in SROMUC can be represented combining such constructs as *actor*, *use case*, *security constraint*; and using such relationships as *communication link (association)*, *includes*, *extends* and *constraints of*. The *supports* relationship of ISSRM between IS-assets and business assets can be represented

using such relationships as *include*, *extend* and *communication (association) link* of SROMUC.

Risk-related concepts of ISSRM in SROMUC can be modeled by combining such constructs as *misuser*, *misuse case*, *impact*, *use case* and *vulnerability*; and linking constructs with such relationships as *communication link*, *exploits*, *negates*, *harms*, *leads to*, *includes* and *extends*. Examples of SROMUC risk-related syntax constructs can be seen in Appendix A: Alignment Tables, Table 5.

Risk treatment-related concepts of ISSRM constructs can be represented using *security use case* and *mitigates* relationship. Examples of SROMUC risk treatment-related syntax constructs can be seen in Appendix A: Alignment Tables, Table 6. The risk treatment and control components of ISSRM can not be modeled using SROMUC.

3.1.2. MagicDraw UML

To model Misuse case diagrams we have chosen Magic Draw UML software. MagicDraw UML is software modeling tool made by NoMagic Inc., it allows to create several types of UML diagrams, including Use case diagrams.

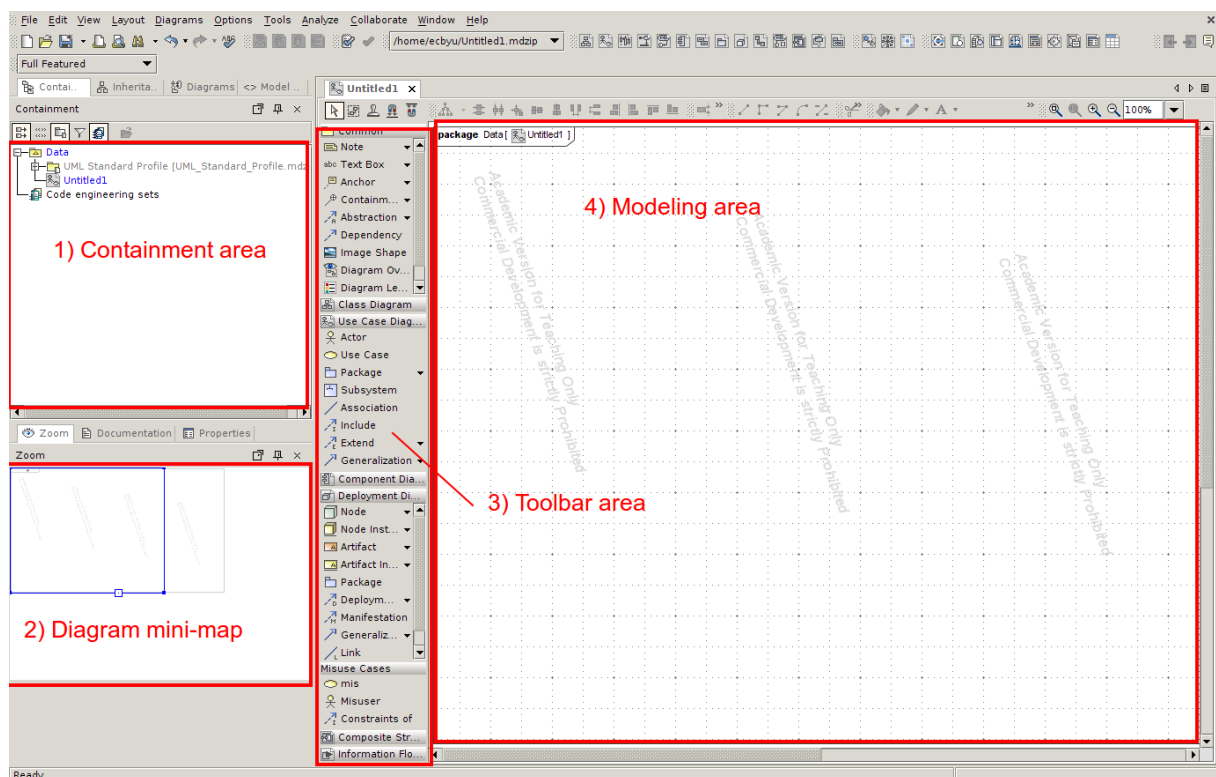


Figure 3: MagicDraw UML user interface

MagicDraw UML has own OpenAPI and JavaDoc materials, allowing users to extend current functionality by producing additional software – plug-ins. The NoMagic company in the interest of the thesis gave us free academical version of the software, which allows us to produce non-commercial plug-ins in academical interests.

MagicDraw UML has simple and user-friendly interface (shown in Figure 3), the main working areas are: (i) containment area – here are shown all elements used in project in the tree view; (ii) diagram mini-map – used for navigating through modeling spaces; (iii) tool-bar area – here are listed all modeling tools; (iv) modeling area – main area, where models are drawn with modeling tools.

The software does not allow to directly model Misuse case diagrams, but it has wide visual and program customization tools, allowing to represent different constructs of Use case models as constructs of Misuse case models (e.g., color filler and applied stereotype functionality). While color filler is clearly visual representation tool – it allows to fill elements with different colors (e.g., filling *actor* with black color visually transforms it to *misuser*), the

applied stereotype functionality can be both: visual and program (e.g, applying “exploits” stereotype to *include* relationship transforms it to *exploits* relationship: visually – instead of “include” sign now is “exploits” sign; and programmatically – software sees applied stereotype “exploits” to *include* relationship).

Examples of modeled Misuse case constructs using MagicDraw UML can be seen in Appendix A: Alignment Tables in Table 4, Table 5 and Table 6. The full documentation for MagicDraw UML can be found at official NoMagic Inc. web-page [Documentation].

3.2. Secure Tropos

Secure Tropos is security oriented development methodology, which is extended from Tropos methodology and i* framework by [Mouratidis and Giorgini, 2007]. It has inherited many elements from Tropos methodology such as *actor*, *goal*, *plan*, *softgoal* and *resource*, adding security-oriented elements such as *security constraint* and *threat*. Secure Tropos is mostly used at early stages of requirement analysis [Mouratidis and Giorgini, 2007]. The further studies [Matulevičius *et al.*, 2012] have extended Secure Tropos and aligned it's constructs with ISSRM.

Secure Tropos is agent-oriented security modeling language, thus the main idea consists in relations between two or more actors [Matulevičius *et al.*, 2012][Mouratidis and Giorgini, 2007]. In Secure Tropos one *actor* (depender) can depend on another *actor* (dependee) through *dependency* relationship and one of the components (*goal*, *resource*, *softgoal*, *plan*). In this case component between two *actors* is called dependum. *Actor* can have own *goals*, *plans*, *softgoals* and use *resources* in order to complete his *goals* or *plans*. The *plan* construct can be decomposed to smaller parts (*goals*, *plans*, *resources*).

3.2.1. Secure Tropos in Terms of ISSRM

In this part we show how Secure Tropos constructs are aligned with ISSRM concepts in three stages: (i) asset-related concepts, (ii) risk-related concepts and (iii) risk treatment-related concepts.

Asset-related concepts of ISSRM in Secure Tropos can be modeled using components: *actor*, *goal*, *resource*, *plan*, *softgoal*, *security constraint*; and relationships: *means-end*, *decomposition*, *contribution*, *satisfies*, *restricts* and *dependency*. As can be seen in Appendix A: Alignment Tables, Table 7, *supports* relationship of ISSRM can be modeled using several relationships of Secure Tropos. The *constraint of* relationship of ISSRM in Secure Tropos is modeled in two ways: implicit and explicit. In implicit relationship dependum of *dependency* relationship is constrained by *security constraint*, showing that constraint is also applied to depender and/or dependee. Explicit relationship is represented by *restricts* relationship and shows that construct (*goal*, *plan*, *resource*) is restricted by *security constraint* [Matulevičius *et al.*, 2012].

Risk-related concepts of ISSRM in Secure Tropos can be modeled using constructs: *actor*, *threat*, *goal*, *plan*, *vulnerability*; and relationships: *means-end*, *decomposition*, *impacts*, *attacks* and *exploits*. Relationships *provokes* and *significance assessed by* of ISSRM can not be modeled using Secure Tropos. The alignment of Secure Tropos constructs to ISSRM risk-related concepts can be seen in Appendix A: Alignment Tables, Table 8.

Risk-treatment concepts of ISSRM can be represented using same components and relationships as were used in modeling asset-related concepts, adding one new relationship – *mitigates*. *Mitigates* relationship can be used by *security constraint* against *risk*. *Risk treatment* construct and relationships *intention to threat* and *refines* of ISSRM can not be modeled using Secure Tropos [Matulevičius *et al.*, 2012]. To see how risk-treatment concepts are modeled using Secure Tropos construct please check Appendix A: Alignment Tables, Table 9.

3.2.2. SecTro2

To model Secure Tropos models we have chosen SecTro2 tool software. SecTro2 is a

software tool made by authors of Secure Tropos methodology [Mouratidis and Giorgini, 2007]. Software tool allows to design and build security oriented models using Tropos and Secure Tropos concept models. SecTro2 is fully free and is available for download at official page of Secure Tropos project after short registration [Secure Tropos].

SecTro2 tool allows to model multilevel Secure Tropos models, using up to five different model views (e.g., Security Requirements View, Security Attacks View, Organizational View). We are interested in modeling only in two views : (i) Security Requirements View – main view, where are mostly modeled asset-related concepts and risk-treatment concepts of Secure Tropos; (ii) Security Attacks View – sub-view for threat elements, where are modeled risk-related concepts of Secure Tropos;

The SecTro2 tool user interface main parts are (Figure 4): (i) model explorer – here are listed all models; (ii) model navigator – mini-map for navigating through model area; (iii) modeling tool-bar – here are listed all tools that are used to model Secure Tropos models; (iv) modeling area – main area, where are models modeled; (v) views – bar with views switchers.

Examples of how Secure Tropos constructs are modeled using SecTro2 tool can be seen in Appendix A: Alignment Tables in Table 7, Table 8 and Table 9. The full user guide for SecTro2 tool can be found at official developers web-page [Secure Tropos].

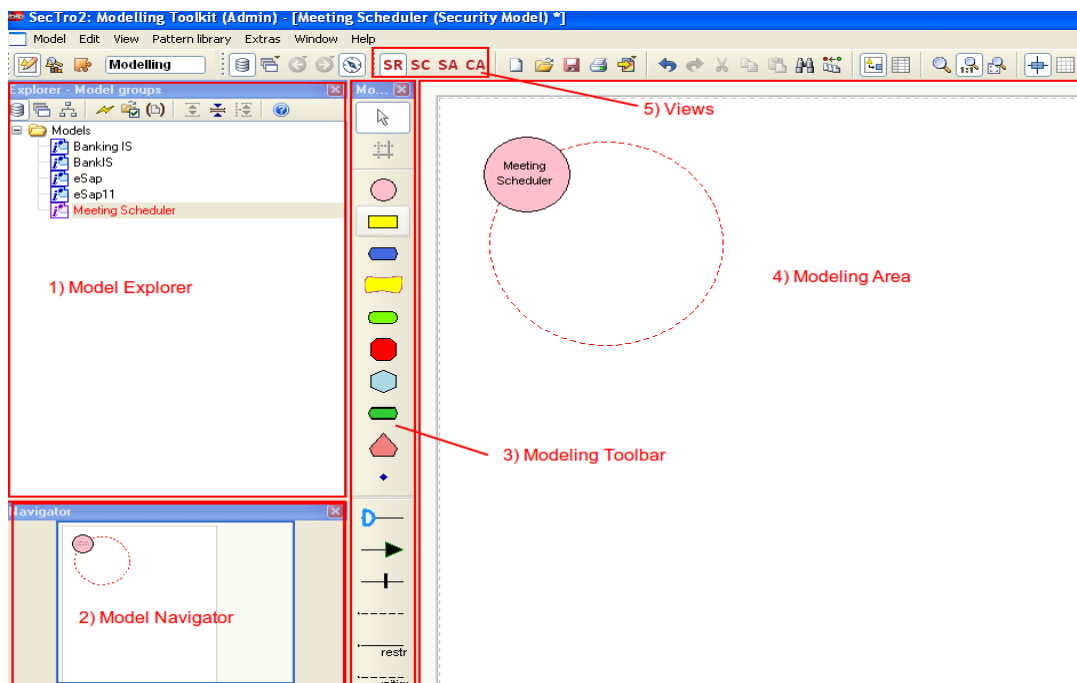


Figure 4: SecTro2 tool user interface

3.3. Summary

The security modeling language is particularistic type of the modeling language, which aims on the modeling and analyzing the security risks of the software system.

Misuse cases is a security modeling language, extended from Use case models. The main idea of Misuse case models is to model the unwanted activity in the software system and propose the security requirements and countermeasures. Misuse cases extended version SROMUC is aligned to all ISSRM domain model concepts in [Soomro and Ahmed, 2013]. Misuse cases can be modeled with MagicDraw UML software.

Secure Tropos is a security modeling language, extended from i* framework and Tropos methodologies and is mostly used at early stages of requirement analysis. The main idea of Secure Tropos consists in relations between two ore more actors. Secure Tropos is aligned to ISSRM domain model in [Matulevičius *et al.*, 2012]. To model Secure Tropos constructs we are using SecTro2 tool.

Chapter 4: Transformation

This chapter introduces a set of transformation rules used in prototype for translating from Misuse case diagrams to Secure Tropos models and *vice versa*. All rules are based on the ISSRM domain model and transformation rules defined in [Ahmed and Matulevičius, 2011] [Soomro, 2012][Ahmed *et al.*, 2012].

4.1. Software Limitations

It is important to understand that software tools, that were used to model and transform between two modeling languages, had limitations in modeling. Limitations are affecting input and output models of prototype, forcing us to make minor changes in transformation rules. The proposed set is made prior to these limitations and may slightly differ from canonical rules. The alignment tables of both security modeling languages in the Appendix A: Alignment Tables are showing how different ISSRM constructs can be modeled using two software tools (MagicDraw UML and SecTro2).

4.2. Transformation Rule Types

Not all transformation rules can be automatically applied by prototype due the software limitations or semantic and syntactic differences between Misuse case diagrams and Secure Tropos models. Transformation rules will be divided by user involvement in transformation process to three types:

1. **Automatic** – transformation rule will be applied by prototype automatically. Prototype is using only elements of diagram to transform between two languages;
2. **Semi-automatic** – prototype needs user defined inputs in order to apply transformation rule. Prototype is using elements of diagram and user inputs to transform between two languages;
3. **Manual** – transformation rule will be not applied by prototype. User needs to apply transformation rules manually after transformation is made by prototype;

4.3. Secure Tropos to Misuse Cases Transformation

This part introduces step-by-step rules applied by prototype to transform from Secure Tropos models to Misuse case diagrams. Transformation rules are based on three concepts: (i) alignment of SROMUC to ISSRM made by [Soomro and Ahmed, 2013]; (ii) previously defined and proposed rules by [Ahmed and Matulevičius, 2011] for transformation from Secure Tropos to MUC; and (iii) the considered software limitations of SecTro2 tool and MagicDraw UML software (see 4.1). The following transformation rules are illustrated only with finished result of Misuse case diagram, which may be confusing, to see how models changed every step please see Appendix B: Transformation Rules Examples.

4.3.1. Model Example

In order to have full understanding of how transformation rules are applied by prototype we will use the slightly changed Secure Tropos model example adapted from [Matulevičius *et al.*, 2012]. Model example was modeled using SecTro2 tool (see Figure 5, Figure 6).

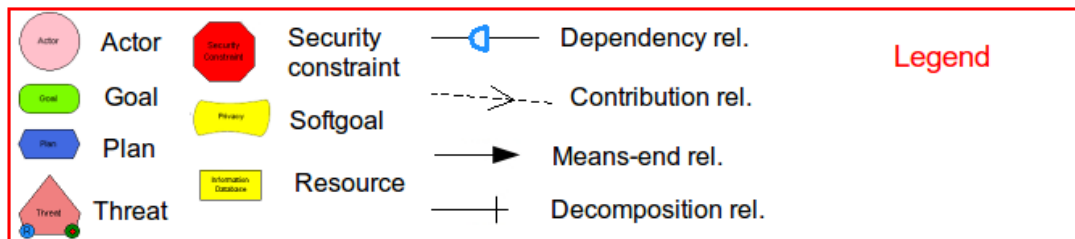
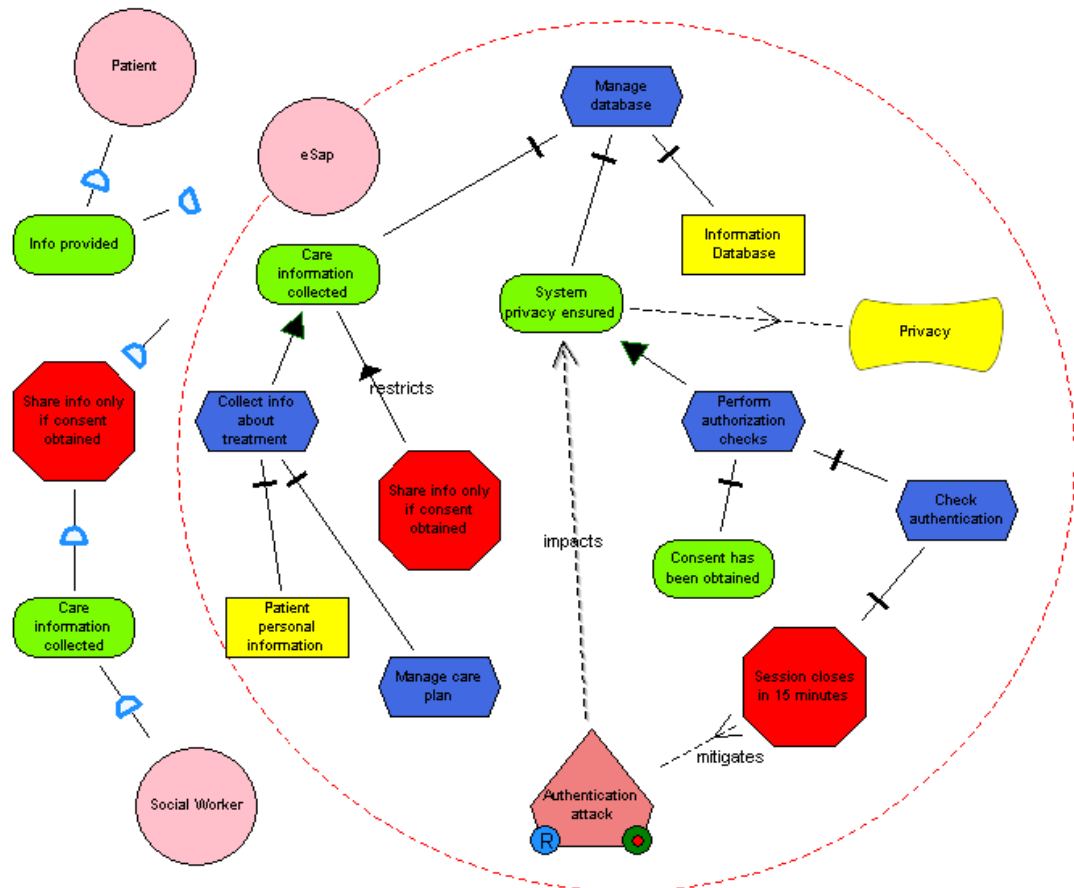


Figure 5: eSap model example modeled with SecTro2 tool, Security Requirements view

SRV - "Authentication attack"

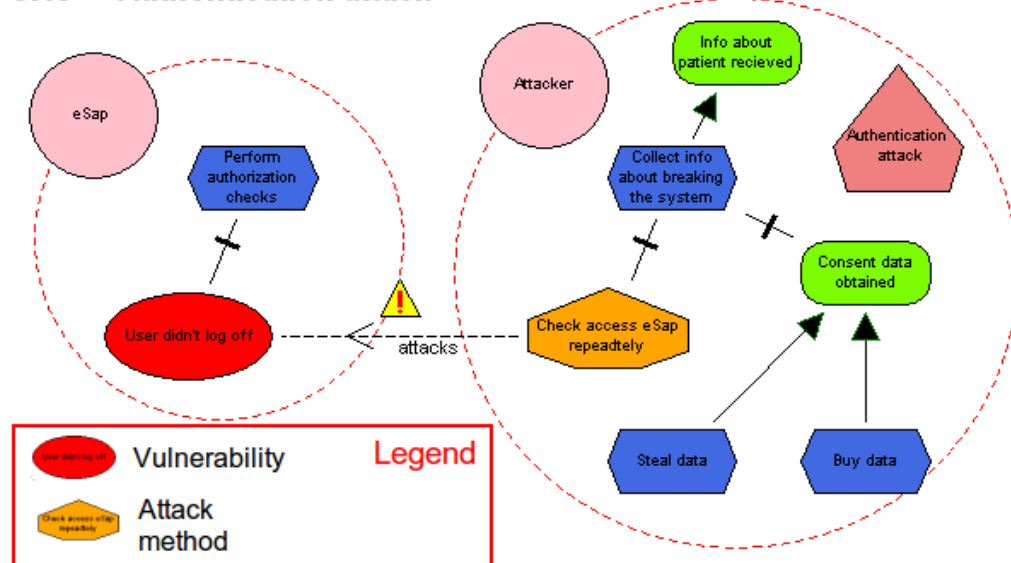


Figure 6: eSap model example "Authentication attack" threat sub-view

As can be seen in Figure 6, the model has three main *actors* cooperating with each other: (i) “eSap” – is IS, designed to store and exchange sensitive patient's information between two other actors – patients and social workers; (ii) “Patient” – is *actor*, whom sensitive information is/can be stored in “eSap” database, thus “eSap” is depending upon the patient in providing this information; (iii) “Social worker” – person who has access to “eSap” and can view stored information and manage patient's care plan only after the consent is provided by patient, thus social worker depends through secure dependency on the “eSap” in providing the sensitive information about patient.

In order to keep sensitive information private “eSap” allows to use and view information only if: (a) social worker is authenticated in system; and (b) consent to share information is obtained. However there exists a threatening *risk* “Authentication attack” with impact on one of the main system's *goals/assets* “System privacy ensured”. The *threat* lies in *vulnerability* of the system's *plan* “Perform authorization checks” (see Figure 5). The *attacker* may just repeatedly check the access to “eSap” and if the *actor* “Social Worker” didn't log off of the system, *attacker* may gain access to patient's sensitive information. In order to mitigate this *risk*, the *security constraint* “Session closes in 15 minutes” is designed and proposed. If there was no activity in last 15 minutes from logged user account the system will automatically close the connection between “eSap” and user.

4.3.2. Transforming Asset-related Concepts

Asset-related concepts will be transformed following this set of the rules:

STMC1. *A software system actor with internal boundaries in Secure Tropos is translated to a software system boundary in SROMUC diagrams. This rule is **automatic**.*

The software system actor “eSap” is translated to a software system boundary (Figure 7).

STMC2. *Secure Tropos actors are transformed to actors in SROMUC. This rule will be applied **automatically**.*

Secure Tropos actors “Patient” and “Social Worker” are translated to SROMUC actors (Figure 7).

STMC3. *Secure Tropos goals and plans are translated as use cases in SROMUC, the relationships between them – means-end and decomposition – are transformed to include relationship of SROMUC diagrams. This rule is **automatic**.*

All goals and plans from Security Requirements view have been translated as *use cases*. Relationships *means-end* and *decomposition* were transformed to *include* relationships (Figure 7).

STMC4. *The dependency link between actors and software system actor in Secure Tropos is translated as communication (association) link connecting actor and associated use case. This rule will be applied **automatically**.*

Dependency relationships between system actor “eSap” and actors “Patient” and “Social Worker” were transformed to *communication(association) links*, connecting actors and associated *use cases* (Figure 7).

STMC5. *Security constraint and restricts relationship of Secure Tropos, that restricts dependency relationship between software system actor and actor are transformed to security criterion and constraints of relationship in SROMUC diagrams. This rule is **automatic**.*

Security constraint “Share info only if consent obtained” was translated as *security criterion*, *restricts relationship* was transformed to *constraints of relationship* (Figure 7).

4.3.3. Transforming Risk-related Concepts

Risk-related concepts will be transformed following this set of the rules:

STMC6. *The actor of Secure Tropos who attacks and exploits the vulnerabilities of system IS-assets (goals or plans) is translated to misuser of SROMUC. This rule is **automatic**.*

The actor from model threat sub-view “Attacker” was transformed to SROMUC misuser (Figure 7).

STMC7. *The goals, plans or attack methods, that belong to actor that exploits or attacks*

vulnerabilities of system IS-assets in Secure Tropos are transformed to misuse cases in SROMUC. The means-end and decomposition relationships of Secure Tropos will be translated to includes relationship. The top element of threat tree construct will be connected via communication link with misuser. This rule will be applied **automatically**.

Threat “Authentication attack” sub-view goals “Info about patient received”, “Consent data obtained”; plans “Collect info about breaking the system”, “Steal data” and “Buy data”; attack method “Check access eSap repeatedly” were transformed to misuse cases. Means-end and decomposition relationships between goals, plans and attack method were translated as include relationships. The communication(association) link was added, connecting misuser “Attacker” and misuse case “Info about patient received” (Figure 7).

STMC8. A Secure Tropos vulnerability transforms to SROMUC vulnerability, Secure Tropos attacks relationship connecting attack method and vulnerability is translated as exploits relationship in SROMUC, moreover the additional connection threatens is made between misuse case representing attack method and use case representing goal or plan with inclusion of attacked vulnerability. This rule is **automatic**.

Vulnerability “User didn’t log off” was translated to vulnerability in SROMUC. Attacks relationship from attack method “Check access eSap repeatedly” towards vulnerability “User didn’t log off” was transformed to exploits relationship. Additional relationship threatens between misuse case “Check access eSap repeatedly” and use case “Perform authorization checks” was added (Figure 7).

STMC9. The impacts relationship of Secure Tropos will be transformed to SROMUC diagram construct which will consist of minimum three elements: (i) leads to relationship – relationship between misuse case representing attack method and impact construct of SROMUC; (ii) impact construct; (iii) harms relationship pointing from impact construct towards the impacted use case representing goal or plan. This rule will be applied **automatically**. The negates relationship between impact construct and security criterion of SROMUC needs to be modeled **manually** by user.

Impacts relationship was translated to SROMUC construct containing two relationships: (i) leads to; (ii) harms; and one element impact “Impact1” (Figure 7).

4.3.4. Transforming Risk Treatment-related Concepts

Risk treatment-related concepts will be transformed following this set of the rules:

STMC10. The security constraint with mitigating relationship towards threat in Secure Tropos will be translated as security use case of SROMUC. This rule is **automatic**.

Security constraint “Session closes in 15 minutes” was translated to security use case (Figure 7).

STMC11. The mitigates relationship of Secure Tropos is transformed as mitigates relationship in SROMUC. This rule will be applied **automatically**.

Mitigates relationship between security constraint “Session closes in 15 minutes” and threat “Authentication attack” was translated to mitigates relationship (Figure 7).

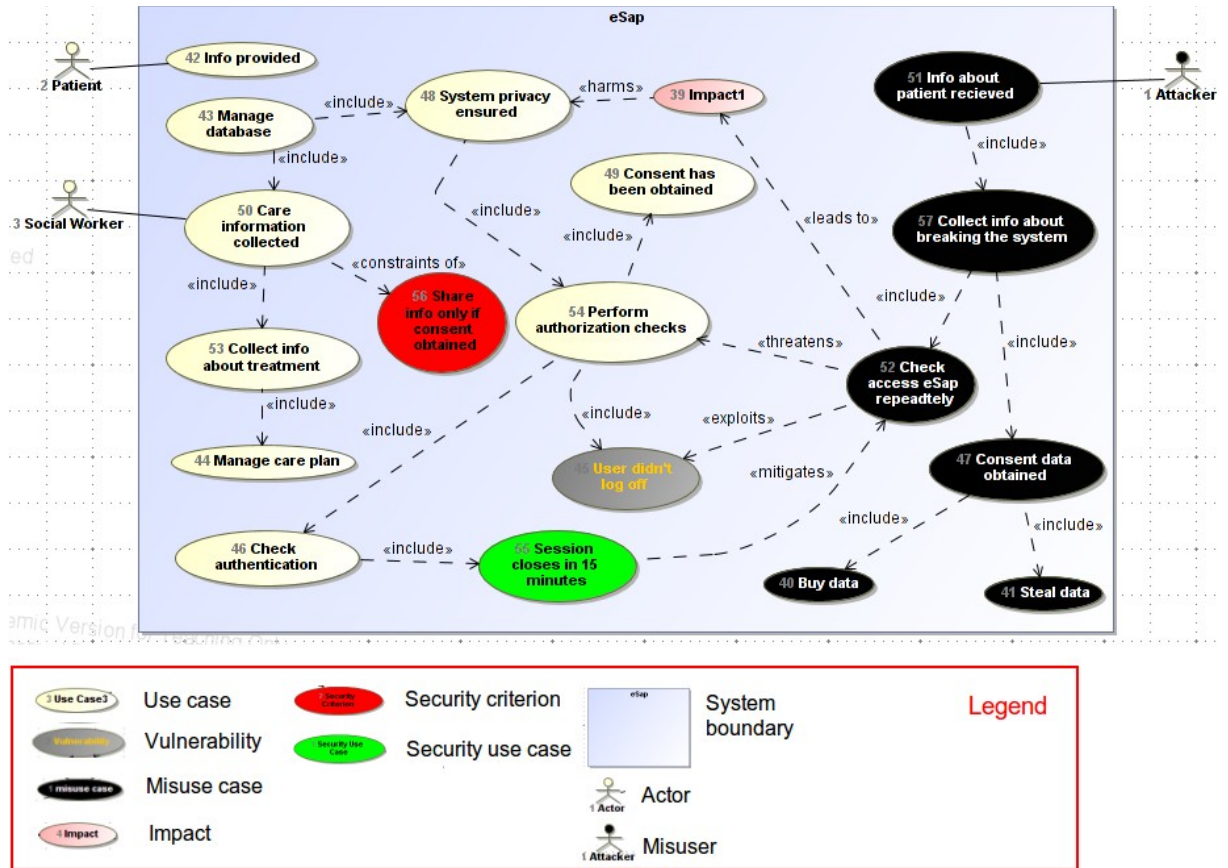


Figure 7: Misuse case result diagram (STMC10 and STMC11 rule example)

4.3.5. Finalizing Transformation

Not all elements and relationships can be translated automatically and in order to finalize transformation from Secure Tropos models to Misuse cases diagrams user has to:

1. Rename *impact* elements. Although *impact* elements are created automatically by prototype, they will be named as “Impact1”, “Impact2”, ..., “ImpactN”; where *N* is amount of *threats* with *impact* relationship in Secure Tropos models;
2. Add *negates* relationship, if it is needed. *Negates* relationship can be used between *impact* and *security criterion* in SROMUC. In Secure Tropos *negates* relationship is represented as *impacts* relationship. In SecTro2 tool it is impossible to connect *threat* element and *goal* or *plan* elements with *impacts* relationship;

4.4. Misuse Cases to Secure Tropos Transformation

This section introduces step-by-step rules applied by prototype to transform from Misuse case diagrams to Secure Tropos models. Transformation rules are based on three concepts: (i) alignment of SROMUC to ISSRM made by [Soomro and Ahmed, 2013]; (ii) previously defined and proposed rules by [Soomro, 2012][Ahmed *et al.*, 2012] for transformation from MUC and SROMUC to Secure Tropos; and (iii) the considered software limitations of SecTro2 tool and MagicDraw UML software (see 4.1). Here we only provide the figures for finished result of Secure Tropos models, which may be confusing, to see how models changed every step please see Appendix B: Transformation Rules Examples.

4.4.1. Diagram Example

To have full understanding how transformation rules are applied by prototype during transformation from Misuse case diagram to Secure Tropos model we will use previously defined Misuse case model (Figure 7).

Transformation from Secure Tropos model to Misuse case diagram have not changed the main ideas of diagram: (i) diagram has one system (“eSap”), represented as system boundary; (ii) diagram has two main actors (“Patient”, “Social Worker”) interacting with “eSap”; and (iii) within diagram is defined *security risk*;

The transformation process from Misuse cases diagrams to Secure Tropos models includes several semi-automatic transformation rules, to apply these rules prototype waits from user specific interaction. In this part we will not discuss the interaction methods with software, here we only provide the transformation rules and its examples.

4.4.2. Transforming Asset-related Concepts

Asset-related concepts will be transformed following this set of the rules:

MCST1. *A system boundary that represents software system in the Misuse case diagram is translated to Secure Tropos system actor with internal goal diagram. This rule is automatic.*

The system boundary “eSap” is transformed to system actor (Figure 9).

MCST2. *A use case from Misuse case diagram is translated either to goal or plan in Secure Tropos model. Includes link is translated either to means-end relationship (goal at the end of the relationship) or to decomposition relationship (plan at the end of the relationship). This rule is semi-automatic; user needs to mark use cases that will be transformed to goals and unmarked use cases will be translated to plans.*

Use cases “Care Information collected”, “System privacy ensured” and “Consent has been obtained” were translated to goals, other use cases were translated to plans. Include relationships between use cases were transformed either to decomposition relations either to means-end relationships (Figure 9).

MCST3. *An actor from the Misuse case diagram is translated to Secure Tropos actor. This rule is automatic.*

Actors “Patient” and “Social Worker” were transformed to Secure Tropos actors (Figure 9).

MCST4. *The security criterion from Misuse case diagrams is transformed to security constraint in Secure Tropos models. “Constraints of” relationship is translated as restricts relationship. This rule is semi-automatic; user defines security criterion and constraints of relationship.*

Security criterion “Share info only if consent obtained” is translated to security constraint. Relationship constraints of was transformed to restricts relation (Figure 9).

MCST5. *Association relationships and associated with them elements from Misuse case diagram will be translated following these rules:*

(i) *If the system actor is dependee, then association relationship will be translated as dependency link in Secure Tropos and use case with which actor is associated to will be translated as dependum (goal either plan) in dependency relationship of Secure Tropos. This rule is semi-automatic; user must identify all actors that will depend upon system actor;*

(ii) *If the system actor is depender in association relationship, then system actor, actor and associated use case must be translated correspondingly. This rule is manual;*

(iii) *If associated use case has “constraints of” relationship with another use case in Misuse case diagram, then use case at the end of the “constraints of” relationship will be included in dependency relationship, acting as security constraint of Security Tropos between system actor and dependum of dependency link (explicit restriction dependency). This rule is semi-automatic; user must define “constraints of” relationship between two use cases and identify security constraints.*

As actor “Social Worker” was marked as depender in dependency relationship with “eSap” system, his communication (association) link with use case “Care information collected” was transformed to dependency relationship. The security constraint “Share info only if consent obtained” was included into dependency relationship, defining explicit restriction dependency relationship between “eSap” and “Social Worker”. Actor “Patient” was not marked as depender, thus his communication link with use case “Info provided” was

semi-automatic; user defines leads to and harms relationships, defines impact element.

The “Impact1” impact element and its relationships leads to and *harms* were transformed to impacts relationship pointing from threat “Threat-1001” element towards goal “System privacy ensured” (Figure 9).

4.4.4. Transforming Risk Treatment-related Concepts

Risk treatment-related concepts will be transformed using this set of the rules:

MCST11. A security use case of Misuse case diagram is transformed to security constraint in Secure Tropos. This rule is *semi-automatic*; user needs to define security use case element.

Security use case “Session closes in 15 minutes” was translated to security criterion (Figure 9).

MCST12. A mitigates relationship from Misuse case diagram is translated to the mitigates link in Secure Tropos. Mitigates relationship can be between security constraint and threat (see MCST7). This rule is *semi-automatic*; user needs to define mitigates link.

The *mitigates* relationship between security use case “Session closes in 15 minutes” and misuse case “Check access eSap repeatedly” was transformed to *mitigates* relationship between security constraint and threat “Threat-1001” (Figure 9).

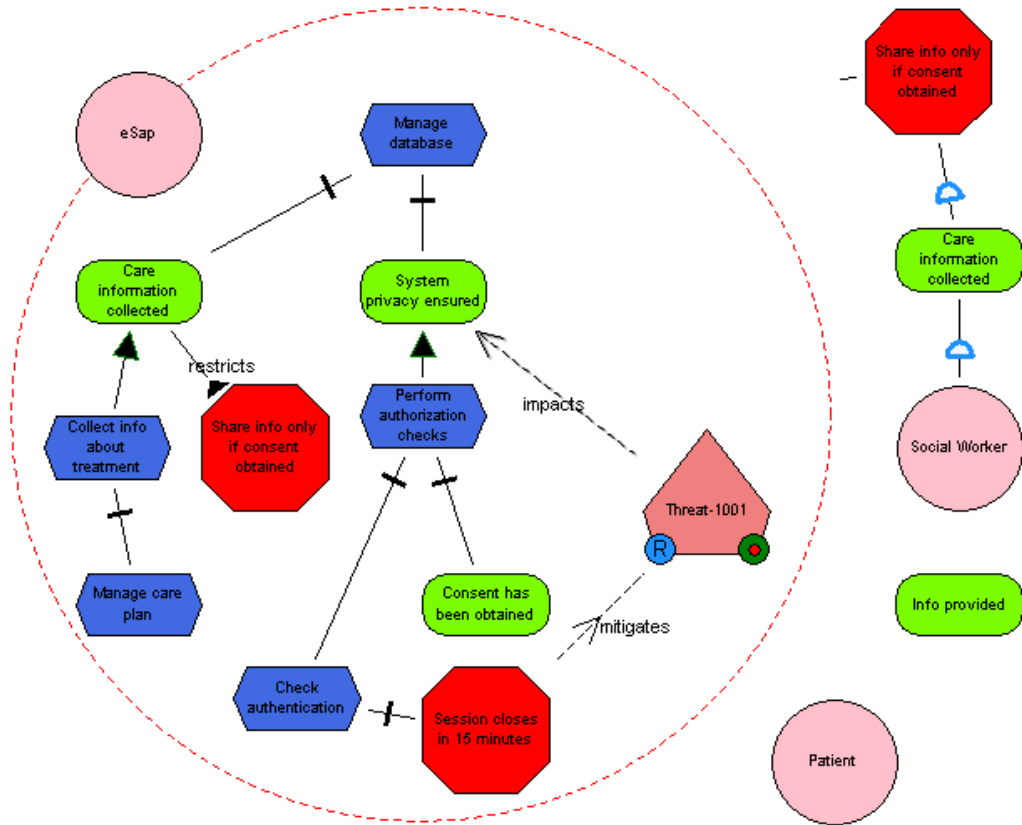


Figure 9: Secure Tropos result model (Security Requirements view) (MCST11 and MCST12 rule example)

4.4.5. Finalizing Transformation

To finalize model transformation from Misuse cases to Secure Tropos user have to:

1. Add missing *dependency* relationships between *actors*. Prototype do not translate *communication (association) links* if *system actor* is *depender*, thus user has to do this **manually**;
2. Add missing elements and relationships. Such elements as *resource* and *softgoal*

are not represented in Misuse case diagrams, thus the rendering of such elements during transformation to Secure Tropos is impossible;

3. Rename all *threat* elements, if it is needed. Prototype automatically names *threat* elements as “*Threat-1001*”,..., “*Threat-(1000+N)*”; where *N* is the amount of *threatens* relationships in Misuse case diagram;

4.5. Summary

In this chapter we introduced the rules to transform from Secure Tropos models to Misuse case diagrams and *vice versa*. The transformation rules are divided by user involvement into transformation process to three different types: (i) automatic; (ii) semi-automatic; and (iii) manual. Prototype uses these rules in order to perform transformation between two security modeling languages. To complete transformation process user has to finalize the transformation in both cases.

Chapter 5: Prototype

This chapter introduces main concept of prototype, discusses it's implementation, code structure, design and requirements. Closer will be discussed the implementation and code structure.

5.1. Design and Requirements

The main concept of prototype is based on the interchange of the information between two software tools, where the Misuse case diagrams (MagicDraw UML) and Secure Tropos models (SecTro2 Tool) are modeled (Figure 10). The information interchange is supported by XML-documents, where are defined only Secure Tropos models. The reason why we use XML file format in exchanging data is very simple – the SecTro2 Tool supports model import and export functionality using XML files, allowing us to use it as a part of the prototype.

We can divide prototype implementation process to two parts: (i) Misuse case transformation to Secure Tropos – information flow from MagicDraw UML to SecTro2 Tool; and (ii) Secure Tropos transformation to Misuse cases – information flow from SecTro2 Tool to MagicDraw UML.

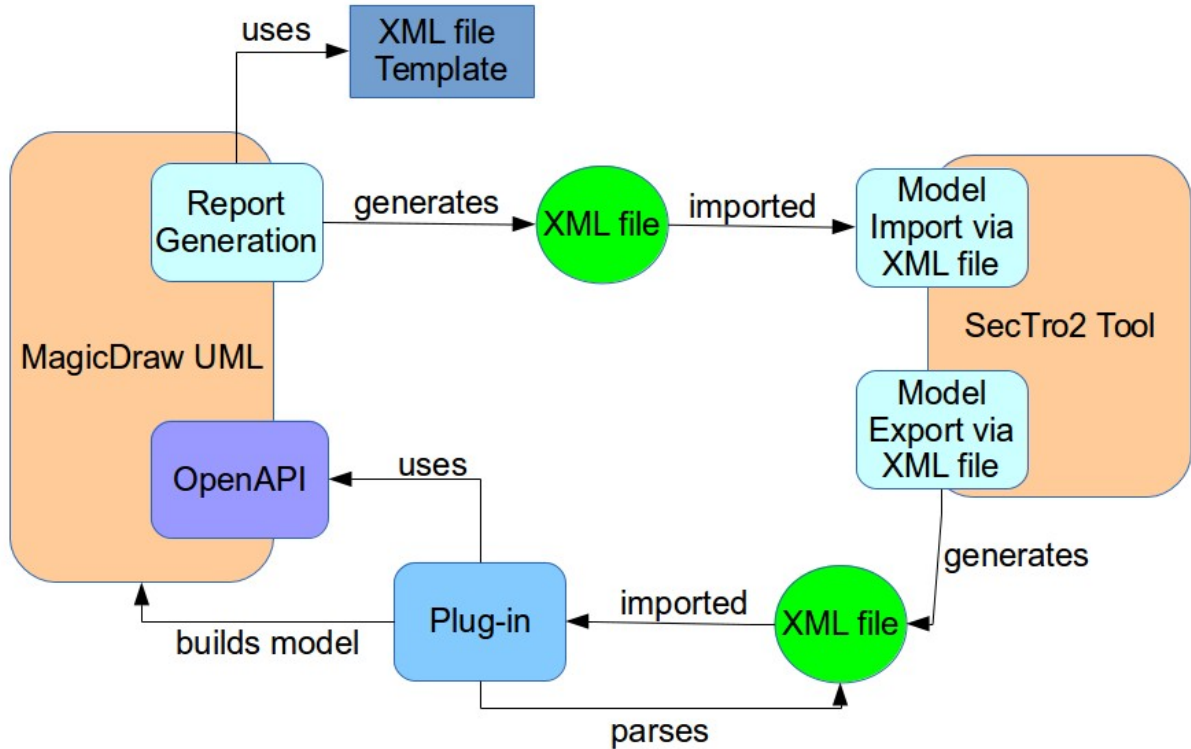


Figure 10: Main concept of the prototype

The functional requirements for both parts of the implementation process are based on two subjects: (i) the transformation process; and (ii) the architecture of the prototype.

5.1.1. Requirements for Misuse Case to Secure Tropos Transformation

In this part we define the functional requirements for Misuse case to Secure Tropos transformation process of the developed prototype.

To transform from Misuse case diagrams to Secure Tropos models we have to generate the XML document, that can be imported to SecTro2 Tool using it's build-in functionality (Figure 10). The XML file has to be similar to original import/export files used in SecTro2 Tool. In this case the XML document needs to contain information about already transformed Secure Tropos model, in other words Misuse case diagram needs to be transformed right before the

XML-document is generated. To generate XML-document we can use the Report Generation functionality of MagicDraw UML.

According to transformation process from Misuse case diagrams to Secure Tropos models (see 4.4) and the MagicDraw limitations over Misuse case diagrams (see 3.1.2) the part of the Misuse case diagram elements and relationships (e.g. *misuse case*, *threatens*) must be predefined by user. Before the transformation process is started user must be able to mark which use cases or misuse cases will be transformed to goals or plans in Secure Tropos.

The list of considered functional requirements based on the written above:

- User must be able to create or use predefined tags/stereotypes in order to differentiate Misuse case additional elements and relationships from Use case elements and relationships;
- User must be able to mark *use cases/misuse cases* to transform them into *goals/plans*;
- User must be able to mark *actors*, that are depending on the system;
- Prototype must be able to read and use user inputs;
- Prototype must transform Misuse case diagram to Secure Tropos before the generation of the XML-document;
- Prototype must be able to generate XML-document using Report Generation functionality of MagicDraw;

In order to meet functional requirements, were proposed next use cases (Figure 11):

MCSTT-1 Input information;

MCSTT-2 Choose which use/misuse cases will transform to goals/plans;

MCSTT-3 Mark depender and dependee;

MCSTT-4 Generate XML file;

MCSTT-5 Use input information;

MCSTT-6 Apply transformation rules;

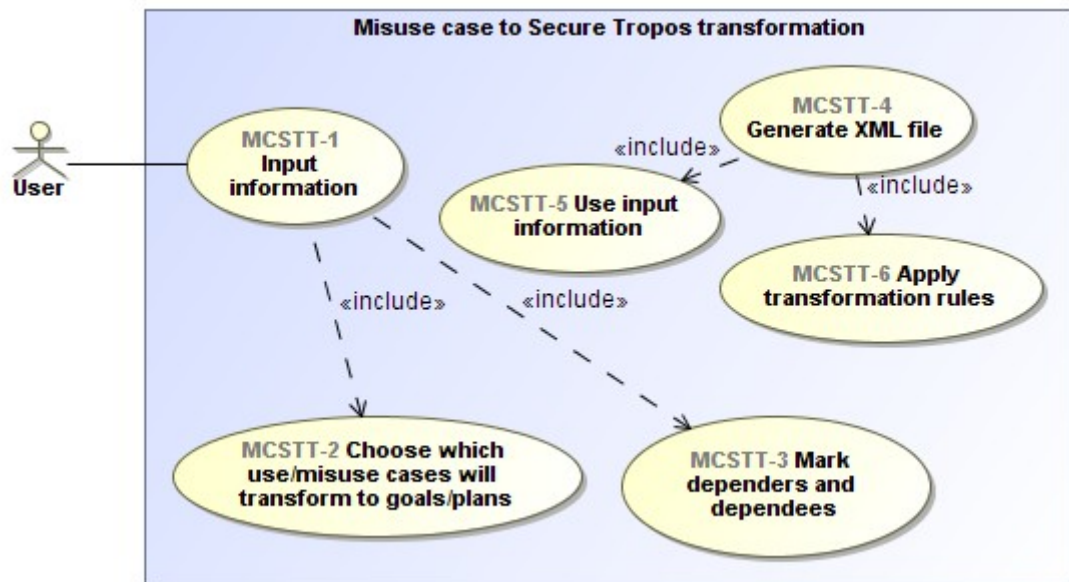


Figure 11: Use cases for Misuse case to Secure Tropos transformation process functional requirements

5.1.2. Requirements for Secure Tropos to Misuse Case Transformation

In this part are defined functional requirements for Secure Tropos to Misuse case transformation process of the developed prototype.

To transform Secure Tropos models to Misuse case diagrams we have to process the

XML-document, that contains information about Secure Tropos model (Figure 10). The MagicDraw UML does not support this functionality, but we can use the OpenAPI of the MagicDraw and define our own plug-in tool, that will parse the XML file.

Another major task in this transformation process is to build the Misuse case diagram from the parsed information. Again, the OpenAPI of MagicDraw will allow us to define the rules of transformation and will help us to build the Misuse case diagram according to transformation rules and parsed information.

According to transformation rules (see 4.3) the user input is not needed directly before the transformation process begins. In our case it will be needed only after transformation process is ended.

The list of considered functional requirements based on the written above:

- User must be able to trigger the XML file input;
- Prototype must be able to parse the XML-document;
- Prototype must be able to build a Misuse case diagram according to parsed information;
- Prototype must be able to create or use predefined tags/stereotypes in order to differentiate Misuse case additional elements and relationships from Use case elements and relationships;
- Prototype must use OpenAPI of MagicDraw UML;

In order to meet these requirements, were proposed next use cases (Figure 12):

STMCT-1 Input XML file;

STMCT-2 Parse XML file;

STMCT-3 Build Misuse case diagram;

STMCT-4 Use parsed information;

STMCT-5 Apply transformation rules;

STMCT-6 Paint Misuse case diagram elements;

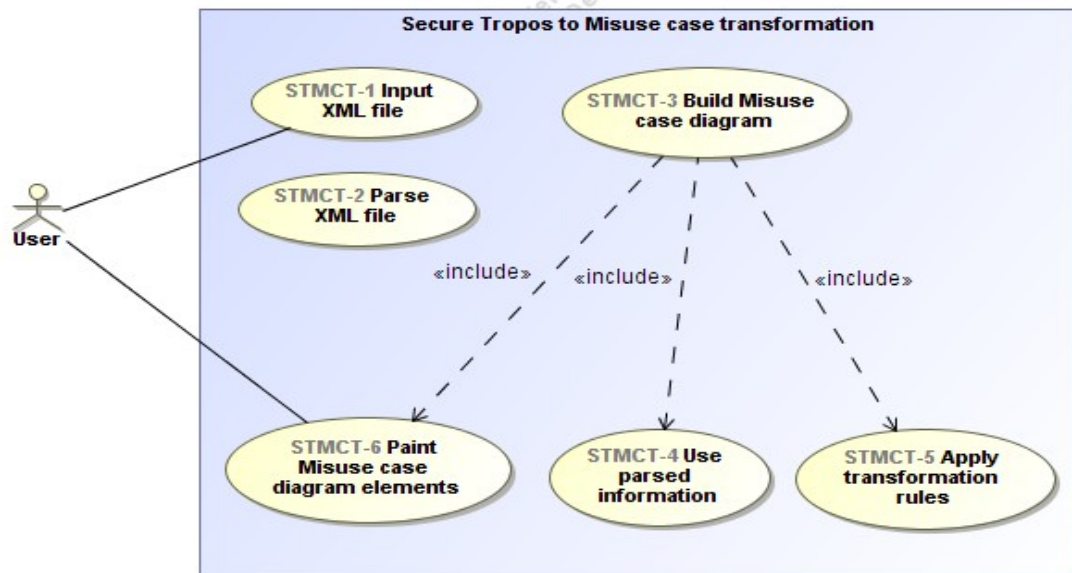


Figure 12: Use cases for Secure Tropos to Misuse case transformation process functional requirements

5.2. Implementation

The implementation process is divided to two independent parts: (i) Misuse case transformation to Secure Tropos; and (ii) Secure Tropos transformation to Misuse cases.

5.2.1. Transformation of Misuse Cases to Secure Tropos

To generate XML-file plug-in tool will be using the Report Generation functionality of MagicDraw. To use Report Generation functionality we defined the XML file template (Figure 10), that contains logics of the transformation rules. The Report Generation is using Apache Velocity Engine and templates are defined using Velocity Template Language. The full user guide for Apache Velocity Template Language can be found at [Velocity]. The user guide for Report Generation Wizard and the template management can be found at official NoMagic web-page [Documentation] in report generation user guide.

The template file can be found in additional CD in Appendix G directory, named as skeleton.xml.

To differentiate the Use case elements and relationships from Misuse case elements and relationships the template file is referencing to stereotypes (see 3.1.2), that are described in Table 1.

Table 1: Misuse case elements stereotype table

Element / Relationship	Actual Element / Relationship	Applied Stereotype
Misuse case	Use case	misuseCase
Vulnerability	Use case	vulnerability
Security criterion	Use case	secConstraint
Impact	Use case	impact
Security use case	Use case	secUseCase
Misuser	Actor	misuser
Constraints of	Include	constraints of
Threatens	Include	threatens
Exploits	Include	exploits
Leads to	Include	leads to
Harms	Include	harms
Negates	Include	negates
Mitigates	Include	mitigates

The user input is realized as template variables – *goals* and *dependers*. To mark which *use cases* are going to be goals in Secure Tropos models user has to add *use case* identification numbers to variable *goals*, joined with semicolon. To mark which *actors* will be dependers in Secure Tropos models user has to add *actors* identification numbers to *dependers* variable, if there are more than one depender, the identification numbers must be joined with semicolon. How variables are used in templates is shown in user guide of Report Wizard, which is located at official NoMagic Inc. web-page [Documentation].

5.2.2. Transformation of Secure Tropos to Misuse Cases

To complete this part of the implementation we developed our plug-in tool, which is based on Open API of the MagicDraw UML software. The main used technology is Java 1.7, as OpenAPI is based on Java programming language.

The plug-in code is separated to three packages (Figure 13): (i) *plugin* – here are placed all Java classes that are directly used by MagicDraw (*e.g.* action triggers, actions, menu configurators); (ii) *plugin.model* – here are placed all classes that will hold the temporary information about parsed instances of XML file (*e.g.* models, elements, relationships); (iii) *plugin.transformation* – here are placed all Java classes, that are used in transformation

process (e.g. model builder, XML parser).

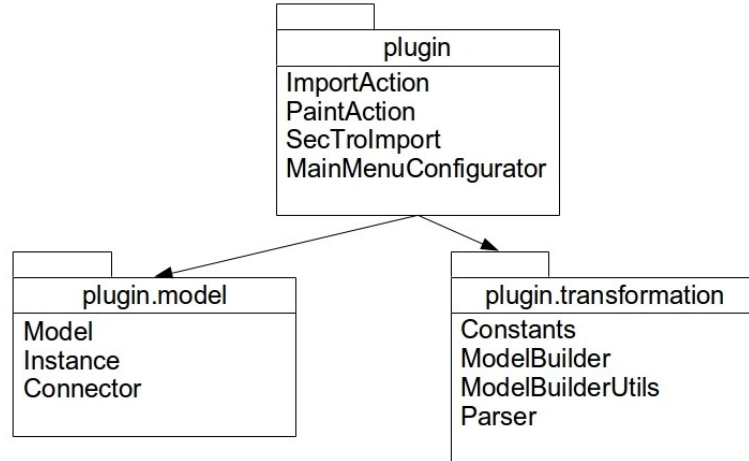


Figure 13: Plug-in package structure

The *plugin* package contains four Java class files (Figure 14): *ImportAction*, *PaintAction*, *SecTroImport* and *MainMenuConfigurator*. The *SecTroImport* class extends the OpenAPI *Plugin* class – it creates buttons in main menu; associates buttons with actions; describes when plug-in is activated and deactivated; holds *MainMenuConfigurator* instance. *MainMenuConfigurator* describes which main menu element is changed by plug-in, in our case it's “Tools” drop-down button in main menu of MagicDraw UML. The *ImportAction* and *PaintAction* are triggers for menu action buttons, they describe what directly happens right after the user clicks on the associated button; and when buttons are active or passive. *ImportAction* class uses the *ModelBuilder* and *Parser* classes of *plugin.transformation* package to: (i) parse the XML document; (ii) save temporary information to *Model*, *Instance* and *Connector* objects; and (iii) build Misuse case model. The action triggers are implemented in *actionPerformed* methods. The *ImportAction* holds temporary models in *modelsToBuild* list, which are *Model* class objects of *plugin.model* package.

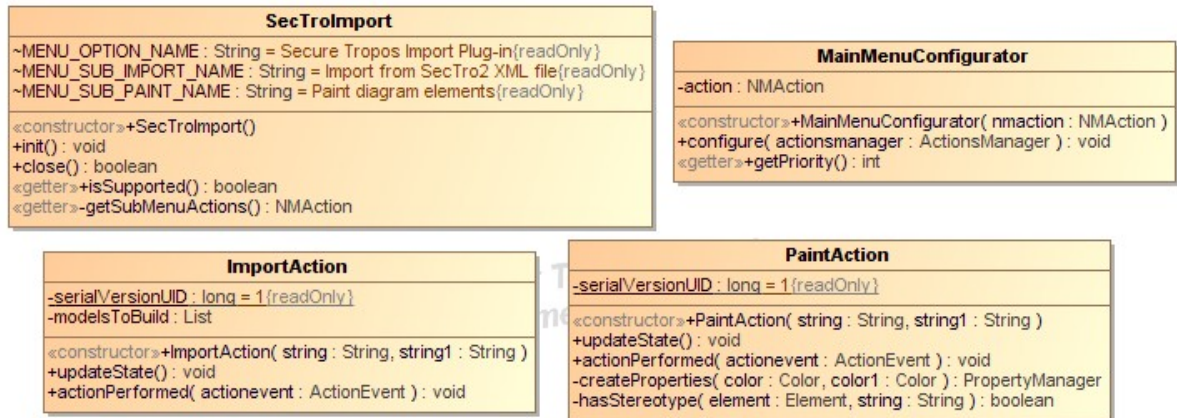


Figure 14: plugin package Java classes

The *plugin.model* package contains three Java classes (Figure 15): (i) *Model* – is the representation of the model that was parsed from XML-document, contains lists of *Instance* and *Connector* objects related to this model; (ii) *Instance* objects are representing the elements that were parsed from imported XML file; (iii) *Connector* objects are holding the information about parsed relationships.

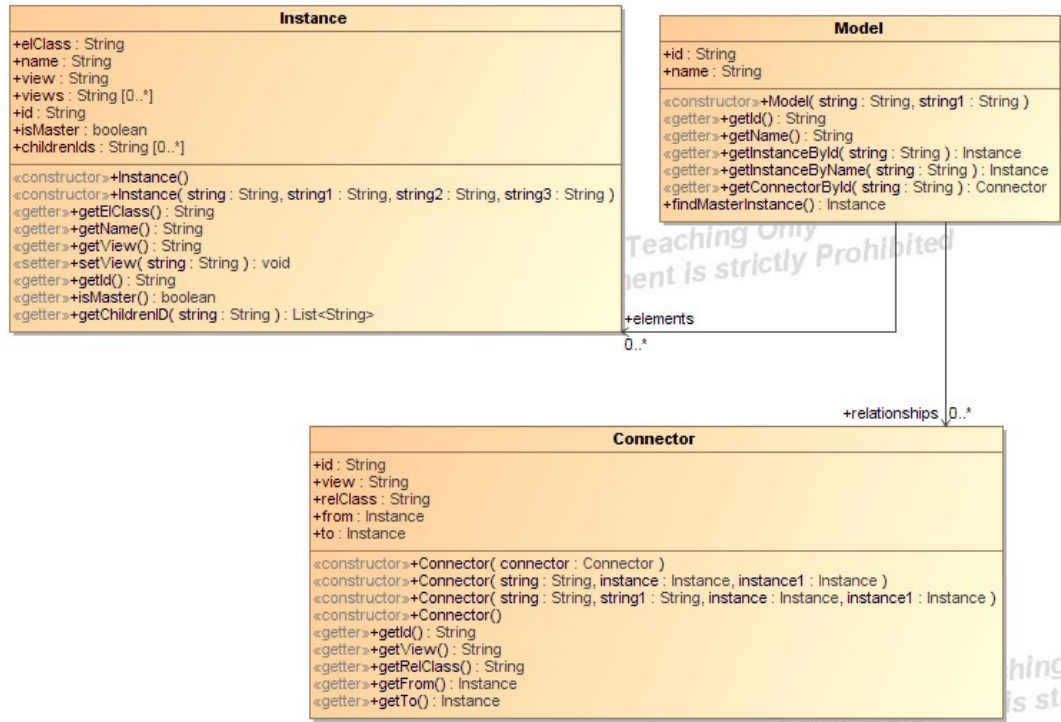


Figure 15: *plugin.model* package Java classes

The *plugin.transformation* package holds four Java classes (Figure 16): (i) *Constants* – holds the constant String elements that are used by *Parser* and *ModelBuilder* classes; (ii) *Parser* - class, that parses the imported XML file, creates *Models*, *Instances* and *Connectors*, applies some transformation rules; (iii) *ModelBuilder* – the actual builder of the Misuse case diagram, applies the bigger part of the transformation rules, creates diagram view, renders shapes of the elements and relationships; (iv) *ModelBuilderUtils* – the helper-class, holds different methods for *ModelBuilder*.

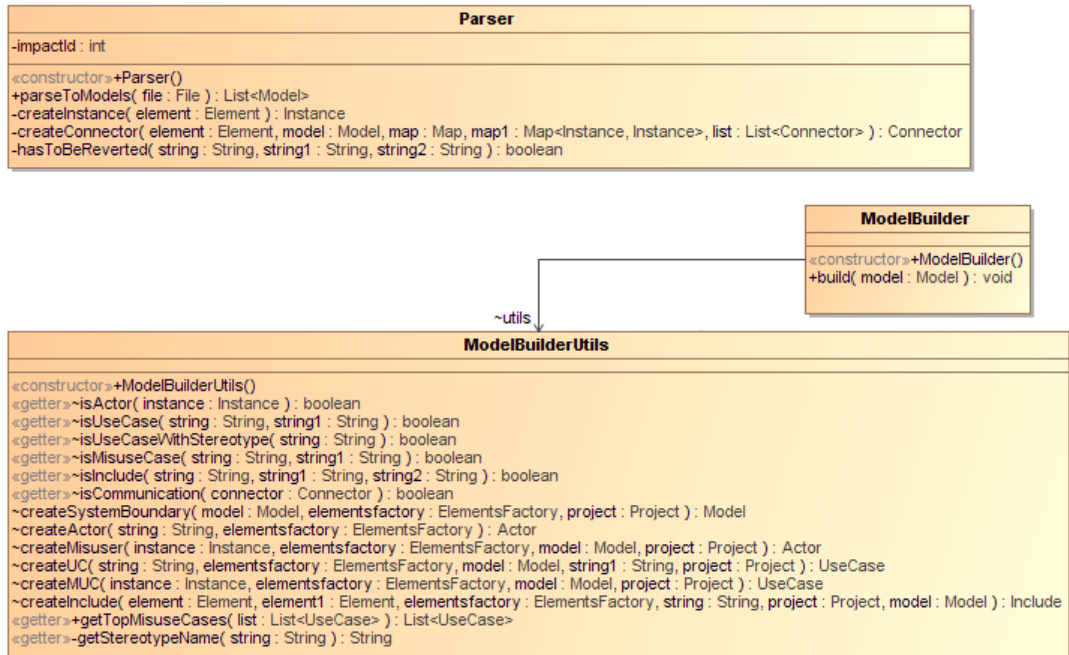


Figure 16: *plugin.transformation* package Java classes

5.3. Summary

The main concept of prototype allows user to share information between two modeling softwares, where the Secure Tropos models (SecTro2) and Misuse case diagrams (MagicDraw UML) are modeled. Prototype uses different functionalities of both modeling softwares in order to complete transformation between two languages. The functional requirements of prototype are based on the main concept and were fully implemented. The implementation of prototype is divided to two parts: Secure Tropos to Misuse case transformation and Misuse case to Secure Tropos Transformation.

Chapter 6: Validation

In this chapter we validate the developed prototype. In order to check validity of the prototype we conduct interview and some practical experiment.

6.1. Scope

Conducting the prototype validation we pursue goals that will demonstrate that the automated transformation process is more easier, efficient (faster) and is preferred over the manual transformation process. According to our goals, the main scope of the validation process is to measure the efficiency and the user friendliness aspects of the developed prototype. The main questions for validation to answer are:

- *Is automated transformation process efficient (faster) than manual transformation process?*
- *Is prototype user friendly and usable?*

The chosen methodologies to answer these questions is a interview and a practical experiment.

In order to validate prototype we have asked five persons, with very low/lack of the experience in software modeling, to take part in the validation process. We believe that software modeling is social activity and people with lack of the experience in modeling will be always involved into it. We believe that prototype could help different types of people, no matter how experienced they are; interviewing these people, we are contributing to our beliefs.

6.2. Validation Process

The validation process is divided to three steps:

1. Theoretical part – we briefly explain the basics of the transformation processes and introduce both languages;
2. Practical part – people are asked to transform certain Secure Tropos model (see Appendix D: Models for Validation) to Misuse case diagram (and *vice versa*) manually (modeling on the paper), then they are asked to transform it using prototype. For each transformation time is noted and written down. During the both processes people can use the transformation rules;
3. Interview – people are asked to answer questionnaire questions. The example of used questionnaire can be seen in Appendix C: Validation Questionnaire;

The questionnaire contains five questions, each question is composed in a way that while user is answering, he has to give the preference to one or another transformation method (automated or manual), in a one to three scale. The greater score is, the greater is the preference of the one transformation method over another.

6.3. Results

The results of the practical experiment can be seen in Table 2. Here are calculated mean times of experiment participants transforming from one modeling language to another manually and automatically. From the automatic process was excluded the software startup time and included time for manual finalization of the diagrams and models (see 4.3.5 and 4.4.5). As can be seen in Table 2, the automatic transformation process is faster in both cases. The difference in time between manual transformations can be explained as follows: in Misuse case to Secure Tropos transformation process experiment participants transformed the same model backwards and they knew what was the final answer. This saved time for transformation. The difference in time between automated transformation processes can be explained with the difference in amount of the work needed to finalize the transformation

process. The automatic transformation on the average took 7 minutes for Secure Tropos to Misuse case transformation and 11 minutes for Misuse cases to Secure Tropos transformation. Greatest part of the time in automatic transformation process was occupied by manual finalization of the result diagrams, whilst the application of automated rules took seconds.

Table 2: Calculated mean time (in minutes) for transformation processes

Transformation process	Manual	Automatic
Secure Tropos to Misuse cases	47 min	7 min
Misuse cases to Secure Tropos	38 min	11 min

The overall results of the interview can be seen in Table 3. As the survey answers had digital representation, we were able to calculate the mean and median for each question and transformation process. To calculate mean and median we represented the preference scores for manual transformation as negative score on -1 to -3 scale; and preference to automatic transformation process as positive score on 1 to 3 scale. As can be seen, in four statements out of five the preference was given to automatic transformation process. With the following statements we will try to analyze and give the explanations to results:

Q1: *Which transformation process was easier to understand?* - Three out of five respondents have given their preference to manual transformation process in this question for both transformation directions. Two respondents gave the biggest available score to manual transformation process. This can be explained with the respondents lack of experience in modeling. While manual transformation was performed by applying transformation rules step-by-step, the automatic transformation applies all transformation rules at once, showing only the final result of transformation.

Q2: *Which transformation process is easier to learn?* - All respondents gave their preference to automated transformation process in both transformation directions. To learn manual transformation process participants would had to remember all rules of transformation and all elements and relationships of two languages. To learn the automated process participants would have to learn the algorithm of using the prototype and smaller part of transformation rules. Obviously the smaller part of transformation rules would be easier to learn.

Q3: *Which transformation process is more efficient (faster) to use?* - All respondents gave the preference to automated transformation process for both transformation directions. Despite the practical experiment had proven that the automation process is faster, the participants didn't knew the results of transformation times, however in their subjective opinion the automatic transformation was still faster than manual process.

Q4: *Which transformation process was/is easier to remember?* - This question differs from Q2 in a way that remembering would mean the availability to repeat the transformation process second time without user manuals or transformation rules. Again, the preference was given to automatic transformation process. The overall preference was given to automatic transformation process. The explanation would be similar to explanation what was used in Q2. The amount of things that had to be remembered is different, thus making automation transformation process more preferable.

Q5: *Which transformation process You would prefer to use in future?* - The overall preference was given to automatic transformation process. The explanation would be very simple – people always tend to use more convenient ways to solve problems. With automatic transformation process participants would only have to remember smaller part of transformation rules, which would be needed for finalization of models. Also the automatic process is faster than manual process.

Table 3: Interview overall results (ST -Secure Tropos; MUC – Misuse cases)

	Q1		Q2		Q3		Q4		Q5	
	ST to MUC	MUC to ST	ST to MUC	MUC to ST	ST to MUC	MUC to ST	ST to MUC	MUC to ST	ST to MUC	MUC to ST
Respondent 1	3	2	3	3	3	3	3	2	3	3
Respondent 2	2	2	3	3	3	3	3	3	3	3
Respondent 3	-2	-2	3	3	3	3	3	3	3	3
Respondent 4	-3	-3	3	3	3	3	3	3	3	3
Respondent 5	-3	-3	3	3	3	3	3	3	3	3
Mean	-0,6	-0,8	3	3	3	3	3	2,8	3	3
Median	-2	-2	3	3	3	3	3	3	3	3

6.4. Threats to Validity

Although the results of validation have showed that prototype is valid, there are existing threats to validity. First, of all the amount of conducted interviews is five, which is very low. The growth of the interviews can readjust the overall picture, and change the validation results. Second, all interviewed had lack of the experience in the modeling, which could affect on the speed of the manual transformation process, and show that automatic process is not faster than manual transformation process. Thirdly – interview and practical experiment was composed and conducted by developer of the prototype. The developer could use only the stronger parts of the prototype and compose the validation in a convenient way that could contribute to positive results of the validation.

6.5. Summary

In order to demonstrate that automated transformation process is easier, more efficient and is preferred over the manual transformation process we conducted the interview and practical experiment. The results of the practical experiment have shown, that automated process is faster. The survey showed that automated transformation process is harder to understand, but the preference in other statements was given to automated process, therefore the prototype is user friendly.

Despite the threads to validity we assume that prototype was validated correctly and results of the validation are correct.

Chapter 7: Conclusion

The aim of the thesis is to define the prototype tool that will automate transformation between two security modeling languages: Secure Tropos and Misuse cases. Both security modeling languages are supported by different modeling softwares: Secure Tropos by SecTro2 tool; and Misuse cases by MagicDraw UML. The transformation between Secure Tropos and Misuse cases is possible due to alignment of both languages to ISSRM domain model and previously defined transformation rules by [Ahmed and Matulevičius, 2011] [Soomro, 2012][Ahmed *et al.*, 2012]. The modeling software tools have limitations and because of that the transformation rules have to be slightly redefined.

The developed prototype allows to transform Secure Tropos models to Misuse case diagrams and *vice versa*, interchanging the data between two modeling softwares, where Secure Tropos models and Misuse case diagrams are modeled. Data interchange is supported with XML-documents. The prototype is separated to two parts, each part is developed independently and may be used individually. The Secure Tropos to Misuse case transformation part is based on OpenApi of the MagicDraw UML and is made using Java programming language. The Misuse case to Secure Tropos transformation part is based on Report Generation functionality of MagicDraw UML and is made using Velocity Template technology [Velocity].

In order to demonstrate that automated transformation process is easier, more efficient and is preferred over the manual transformation process was conducted validation process. The validation showed that automated process is faster, easier and more preferable over manual transformation process.

7.1. Limitations

The developed prototype has several limitations. Firstly, prototype uses slightly changed transformation rules due to limitation of modeling softwares, therefore the prototype transformation rules can differ from proposed rules in [Ahmed and Matulevičius, 2011] [Soomro, 2012][Ahmed *et al.*, 2012]. The prototype takes in account only Misuse case diagrams, the Misuse case templates were not considered. The implementation of the prototype can be improved with refactoring of the source code and reimplementation of the Misuse case to Secure Tropos transformation part. The validation process involved limited number of respondents, which may

7.2. Conclusions

The main goal of the thesis is achieved – the prototype is completed. With our work we raised two questions: (i) How to manage security risks using different modeling languages and to keep model consistency? and (ii) How to automate model transformation between Secure Tropos and Misuse cases? To answer first question we analyzed the ISSRM domain model and transformation rules between Secure Tropos and Misuse cases. To answer second question we developed and validated prototype. By developing prototype we showed that automated transformation between Secure Tropos and Misuse cases is possible. The prototype also contributes to answering first question since it keeps the consistency of the model while transforming from one language to another.

7.3. Future Work

The limitations are showing that there still remains future work with prototype. The work is mostly related with improvement development. The improvement developing could solve the code refactoring problem and redefine transformation rules if the modeling software limitations will be resolved with new versions of the modeling softwares. The validation of the prototype can be redone, now with the greater amount of respondents.

Bibliography

- [Ahmed and Matulevičius, 2011] Ahmed N. and Matulevičius R. “Towards Transformation Guidelines from Secure Tropos to Misuse Cases”. In *Proceeding of the 7th international workshop on Software engineering for secure systems*: (Eds.) J. Jürjens, DOI 10.1145/1988630.1988638, pp. 36-42, ACM, 2011
- [Ahmed *et al.*, 2012] Ahmed N., Matulevičius R. and Mouratidis H. “A Model Transformation from Misuse Cases to Secure Tropos”. In *Proceedings of the CAiSE'12 Forum*: (Eds.) M. Kirikova and J. Stirna, vol. 855, ISBN 1613-0073, pp. 7-14, CEUR-WS.org, 2012
- [Documentation] NoMagic Documentation. 2014. *Documentation*. (ONLINE) Available at: <http://www.nomagic.com/support/documentation.html>. (Accessed 08 May 2014).
- [Dubois *et al.*, 2010] Dubois E., Heymans P., Mayer N. and Matulevičius R. “A Systematic Approach to Define the Domain of Information System Security Risk Management”. In *Intentional Perspectives on Information System Engineering*: (Eds.) S. Nurcan, DOI 10.1007/978-3-642-12544-7_16, pp. 289-306, Springer Berlin Heidelberg, 2010
- [Lodderstedt *et al.*, 2002] Lodderstedt T., Basin D. and Doser J. “SecureUML: A UML-Based Modeling Language for Model-Driven Security”. In *«UML» 2002 — The Unified Modeling Language*: (Eds.) J.M. Jézéquel, DOI 10.1007/3-540-45800-X_33, pp. 426-441, Springer Berlin Heidelberg, 2002
- [Matulevičius *et al.*, 2012] Matulevičius R., Mouratidis H., Mayer N., Dubois E. and Heymans P. “Syntactic and Semantic Extensions to Secure Tropos to Support Security Risk Management”. In *Journal of Universal Computer Science*, vol. 18 (6), ISSN 0948-695x, pp. 816-844, 2012
- [Mayer, 2009] Mayer N. “Model-based Management of Information System Security Risk”, Ph. D thesis, University of Namur, 2009
- [Mouratidis and Giorgini, 2007] Mouratidis H. and Giorgini P. “Secure Tropos: A Security-Oriented Extension of the Tropos Methodology”. In *International Journal of Software Engineering and Knowledge Engineering*, vol. 17 (2), pp. 285-309, 2007
- [Secure Tropos] Secure Tropos | Secure Software Development Methodology. 2014. *Secure Tropos | Secure Software Development Methodology*. (ONLINE) Available at: <http://www.securetropos.org/>. (Accessed 08 May 2014).
- [Silver, 2009] Silver B. “BPMN Method and Style: A levels-based methodology for BPM process modeling and improvement using BPMN 2.0”, Cody-Cassidy Press, 2009
- [Sindre and Opdahl, 2005] Sindre G. and Opdahl A. L. “Eliciting Security Requirements with Misuse Cases”. In *Requirements Engineering Journal*, vol. 10, DOI 10.1007/s00766-004-0194-4, pp. 34-44, Springer-Verlag, 2005
- [Sindre, 2007] Sindre G. “Mal-Activity Diagrams for Capturing Attacks on Business Processes”. In *Requirements Engineering: Foundation for Software Quality*: (Eds.) P. Sawyer, DOI 10.1007/978-3-540-73031-6_27, pp. 355-366, Springer Berlin Heidelberg, 2007
- [Soomro, 2012] Soomro I. “Alignment of Misuse Cases to ISSRM”, Master's thesis, University of Tartu, 2012
- [Soomro and Ahmed, 2013] Soomro I. and Ahmed N. “Towards Security Risk-oriented Misuse Cases”. In *Business Process Management Workshops*: (Eds.) M. La Rosa and P.

Soffer, DOI 10.1007/978-3-642-36285-9_68, pp. 689-700, Springer Berlin Heidelberg, 2013

[Velocity] Apache Velocity - Velocity User Guide. 2014. *Apache Velocity - Velocity User Guide*. (ONLINE) Available at: <https://velocity.apache.org/engine/devel/user-guide.html>. (Accessed 08 May 2014).

Appendices

Appendix A: Alignment Tables

Table 4: SROMUC asset-related constructs in terms of ISSRM (adapted from [Soomro and Ahmed, 2013])

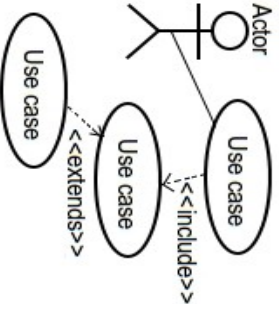
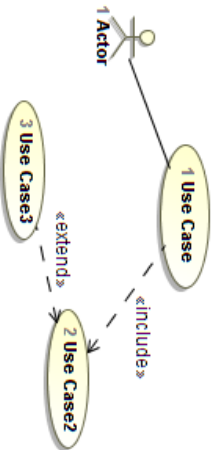
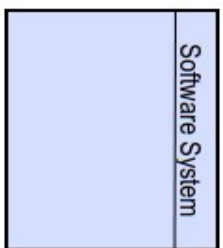
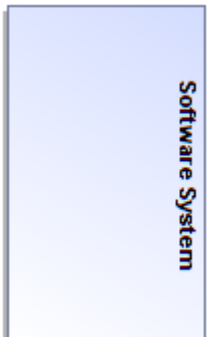
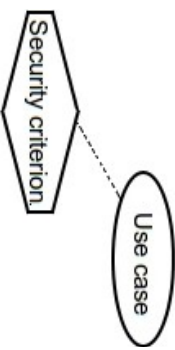

ISSRM Domain Model (C-Construct, R-Relationship)		SROMUC Syntax	Comments	MagicDraw UML Example
Asset	C		<i>Actor can communicate with use case through communication link. Use cases can include one or many use cases and can be extended with another use case.</i>	
Business asset	C			
Supports	R			
IS-asset	C		IS-assets can be represented as whole software system with system boundaries or as one or several use cases connected to each other with <i>include</i> or <i>extends</i> relationships.	
Security criterion	C		<i>Use case can be constrained by security criterion, using constraints of relationship.</i>	
Constraints of	R			

Table 5: SROMUC risk-related constructs in terms of ISSRM (adapted from [Soomro and Ahmed, 2013])

ISSRM Domain Model (C-Construct, R-Relationship)		SROMUC Syntax	Comments	MagicDraw UML Example
Risk	C		<p>Risk in SROMUC is a combination of <i>misuser</i>, <i>misuse case</i>, <i>impact</i>, <i>use case</i> and <i>vulnerability</i>. Risk leads to <i>impact</i>. <i>Impact</i> may harm <i>IS-assets</i> or <i>negate</i> the security criterion.</p>	
Leads to	R			
Harms	R			
Negates	R			
Impact	C		<p><i>Impact</i> may harm one or more assets (<i>use cases</i>) and <i>negate security criterion</i>.</p>	
Threat	C		<p>Threat is a combination of <i>misuser</i>, one or more <i>misuse cases</i>, <i>communication link</i>, <i>include</i> or <i>extends</i> relationships.</p>	
Vulnerability	C		<p><i>Use case</i> can include <i>vulnerabilities</i> and <i>misuse case</i> can exploit it while <i>threatens</i> same <i>use case</i>.</p>	
Event	C		<p>Event can be modeled using <i>misuser</i>, <i>use case</i>, <i>vulnerability</i> and <i>misuse case</i>. Event is risk without <i>impact</i>.</p>	
Exploits	R			

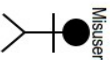



Threat agent	C		Threat agent is represented as negative entity of actor - misuser.	
Attack method	C		Misuse case can threaten use case, while exploiting vulnerability; representing an attack method.	

Table 6: SROMUC risk treatment-related constructs in terms of ISSRM (adapted from [Soomro and Ahmed, 2013])


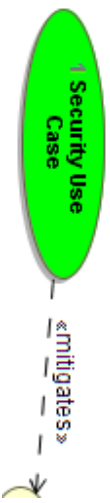
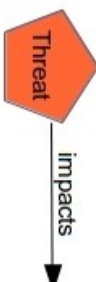
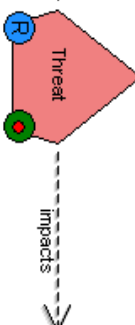

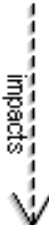






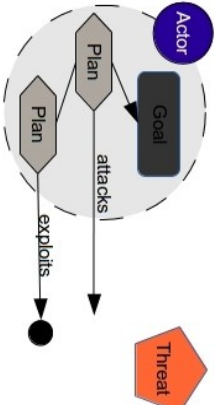
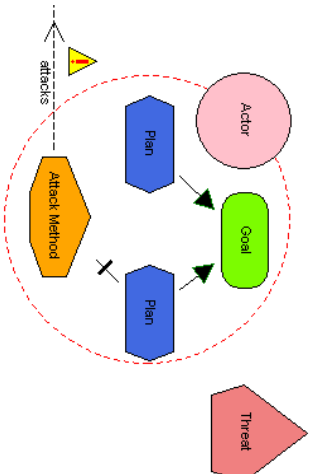
ISSRM Domain Model (C-Construct, R-Relationship)		SROMUC Syntax	Comments	MagicDraw UML Example
Security requirement	C		Security use case can mitigate misuse case and be negated with risk impact.	
Mitigates	R			

Table 7: Secure Tropos constructs in terms of asset-related concepts of ISSRM (adapted from [Matulevičius et al., 2012])

ISSRM Domain Model (C-Construct, R-Relationship)		Secure Tropos Constructs	Secure Tropos Components and Syntax	SecTro2 Tool Example
Asset	C	Assets are represented as a combination of <i>actors</i> , <i>goals</i> , <i>plans</i> , <i>resources</i> and <i>softgoals</i> .		
Business asset	C			
IS-asset	C			
Supports	R	Supports relationship can be modeled using: <i>means-end</i> , <i>decomposition</i> , <i>contribution</i> , <i>satisfies</i> and <i>dependency</i> relationships.		
Security criterion	C	<i>Softgoal</i> , <i>security constraint</i> , <i>decomposition</i> of <i>security constraints</i> , <i>security constraint</i> contribution to <i>softgoal</i>		
Constraint of	R	Implicit <i>constraint of relationship</i> is used in dependency relationship as second dependum. Explicit <i>constraint of relationship</i> is modeled using <i>restricts</i> relationship.		

Table 8: Secure Tropos risk-related constructs in terms of ISSRM (adapted from [Matulevičius et al., 2012])

ISSRM Domain Model (C-Construct, R-Relationship)		Secure Tropos Constructs	Secure Tropos Components and Syntax	SecTro2 Tool Example
Risk	C	<i>Threat combined with impacts relationship</i>		
Impact	C	<i>Impacts relationship</i>		
Leads to	R			
Harms	R			
Negates	R			
Threat	C			
Uses	R	<i>Goal or plan Actor with plan or goal</i>		
Vulnerability	C	<i>Vulnerability can't be modeled directly, but it can be identified as attribute of asset.</i>		
Characteristic of	R			
Exploits	R	<i>Exploits relationship</i>		
Event	C	1) The combination of actor, goals, plans, vulnerability, and decomposition, means-end relationships. 2) Threat		



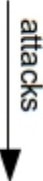

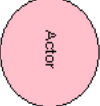

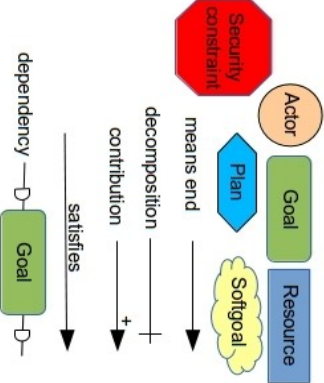
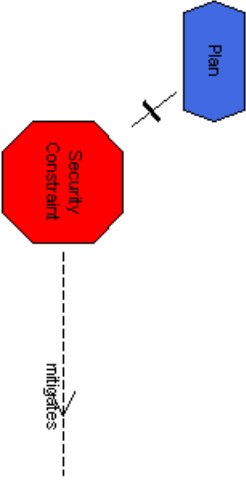
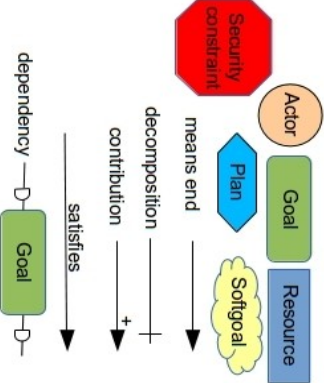
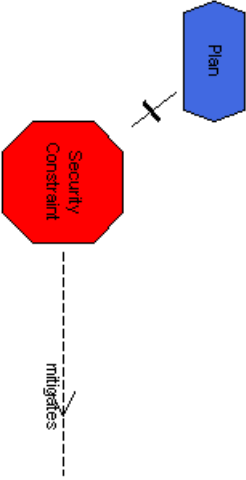
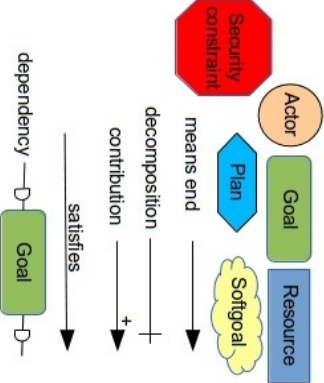
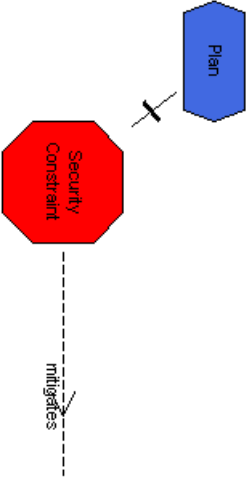
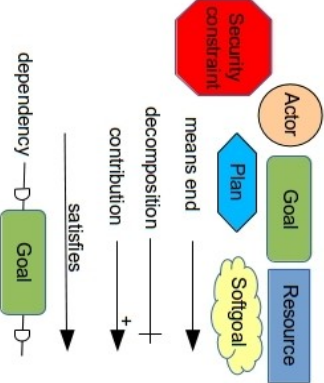
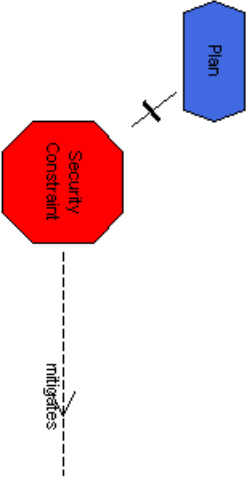
Threat agent	C	<i>Actor</i>		
Attack method	C	Plan, Attack Method		
Targets	R	<i>Attacks</i> relationship		
				
				

Table 9. Secure Tropos risk-treatment constructs in terms of ISSRM (adapted from [Matulevičius et al., 2012])

ISSRM Domain Model		Secure Tropos Constructs		Secure Tropos Components and Syntax		SecTro2 Tool Example	
<i>(C-Construct, R-Relationship)</i>							
Security requirement	C	<i>Actor, goal, resource, plan, softgoal, security constraint.</i>					
Control	C	Combination of components (on the right) and means-end, decomposition, contribution, satisfies, dependency relationships.					
Mitigates	R	<i>Mitigates</i> relationship					
Implements	R	Implicitly in the process of modeling					

Appendix B: Transformation Rules Examples

Table 10: Examples of Secure Tropos constructs transformed to Misuse case constructs

Rule ID	Secure Tropos construct (before)	Misuse case construct (after)	Figure
Asset-related concepts			
STMC1	<i>System Actor</i>	<i>System boundary</i>	Figure 17
STMC2	<i>Actor</i>	<i>Actor</i>	Figure 18
STMC3	<i>Goal</i>	<i>Use case</i>	Figure 19
STMC3	<i>Plan</i>	<i>Use case</i>	Figure 19
STMC3	<i>Means-end and decomposition relationships between plans and goals</i>	<i>Include relationships between use cases</i>	Figure 19
STMC4	<i>Dependency relationships between actors</i>	<i>Communication (association) links between actors and associated use cases</i>	Figure 20
STMC5	<i>Security constraint</i>	<i>Security criterion</i>	Figure 21
STMC5	<i>Restricts relationship</i>	<i>Constraints of relationship</i>	Figure 21
Risk-related concepts			
STMC6	<i>Actor</i>	<i>Misuser</i>	Figure 22
STMC7	<i>Goal</i>	<i>Misuse case</i>	Figure 23
STMC7	<i>Plan</i>	<i>Misuse case</i>	Figure 23
STMC7	<i>Attack method</i>	<i>Misuse case</i>	Figure 23
STMC7	<i>Means-end and decomposition relationships between plans, goals and attack methods</i>	<i>Include relationships between misuse cases. Communication (association) link between misuser and top misuse case.</i>	Figure 23
STMC8	<i>Vulnerability</i>	<i>Vulnerability</i>	Figure 24
STMC8	<i>Attacks relationship from attack method towards vulnerability</i>	<i>Exploits relationship from misuse case towards vulnerability, threatens relationship from misuse case towards use case</i>	Figure 24
STMC9	<i>Impacts relationship</i>	<i>Construct of leads to, harms relationships and impact element</i>	Figure 25
Risk treatment-related concepts			
STMC10	<i>Security constraint with mitigates relationship</i>	<i>Security use case</i>	Figure 7
STMC11	<i>Mitigates relationship</i>	<i>Mitigates relationship</i>	Figure 7

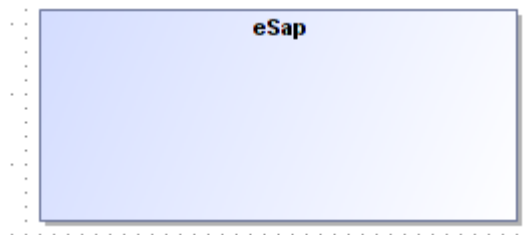


Figure 17: STMC1 rule example



Figure 18: STMC2 rule example

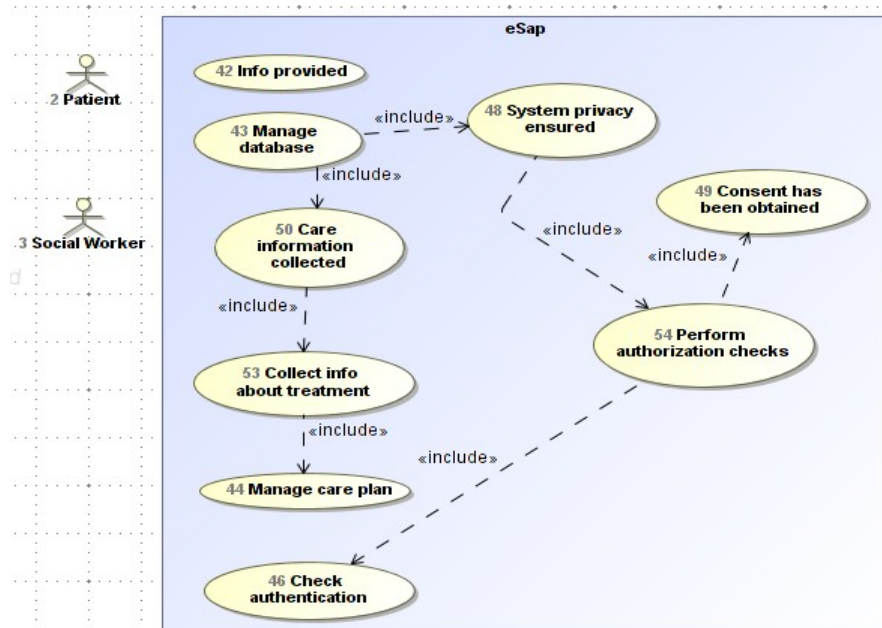


Figure 19: STMC3 rule example

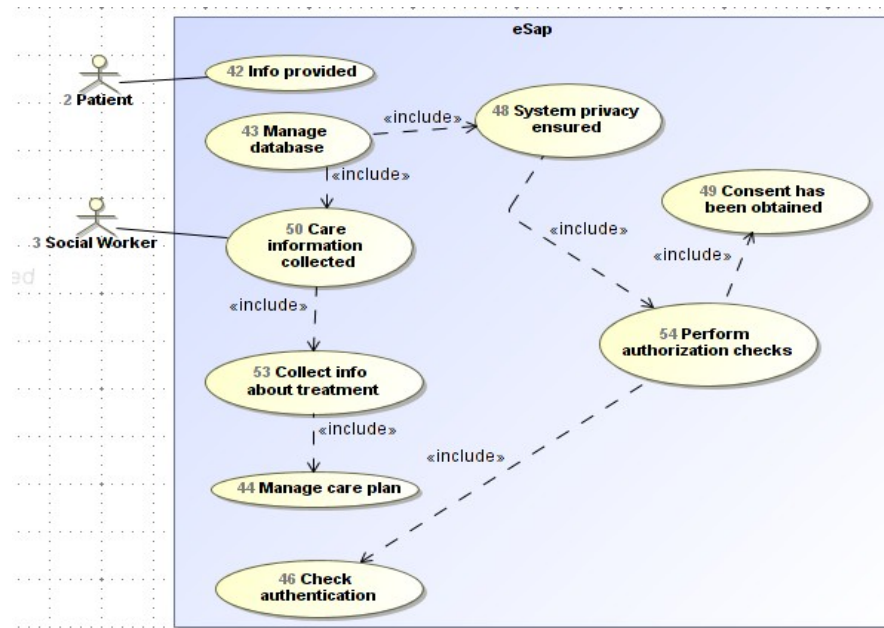


Figure 20: STMC4 rule example

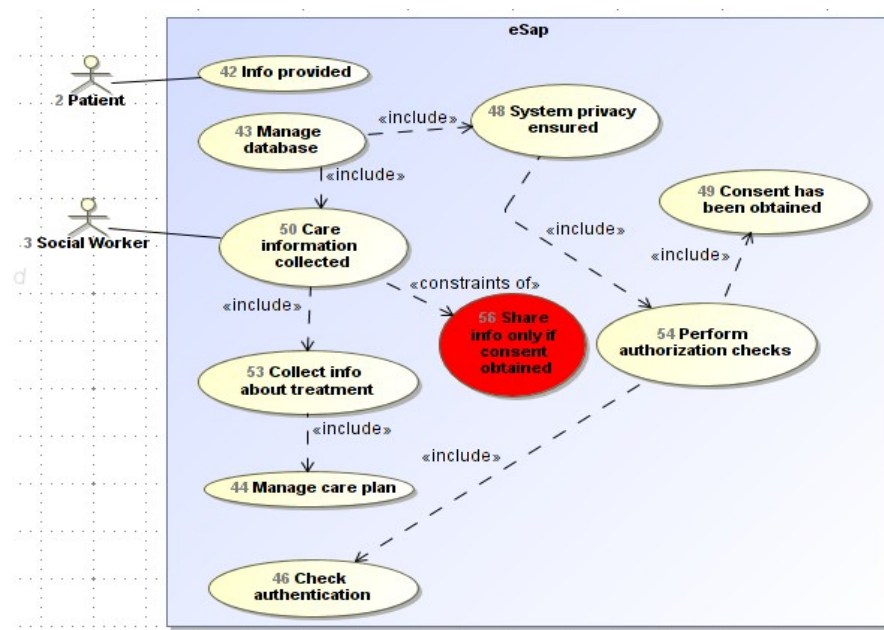


Figure 21: STMC5 rule example

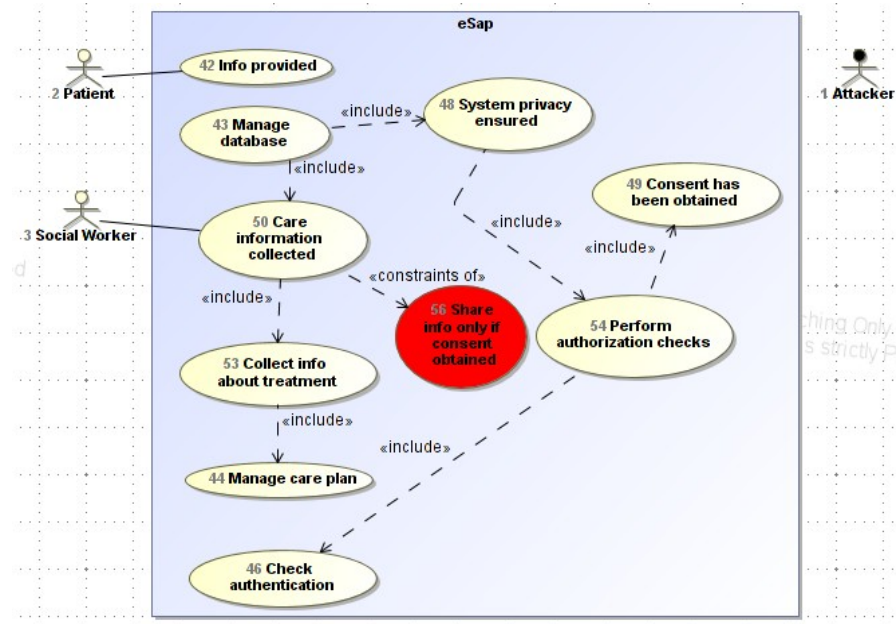


Figure 22: STMC6 rule example

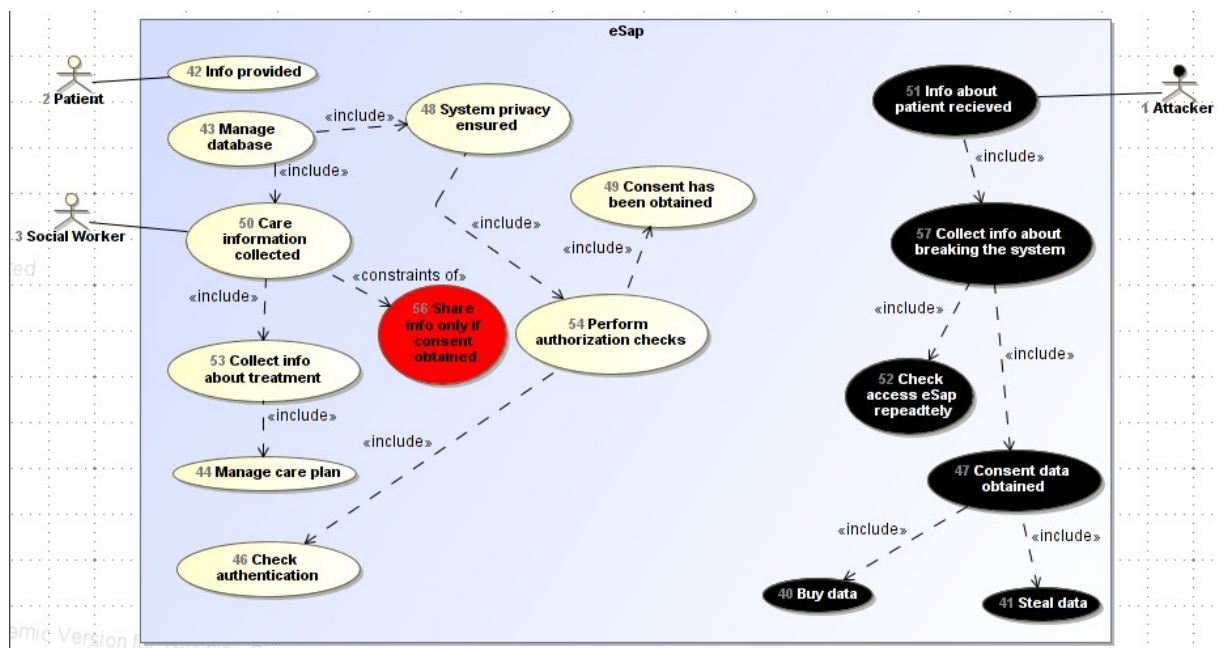


Figure 23: STMC7 rule example

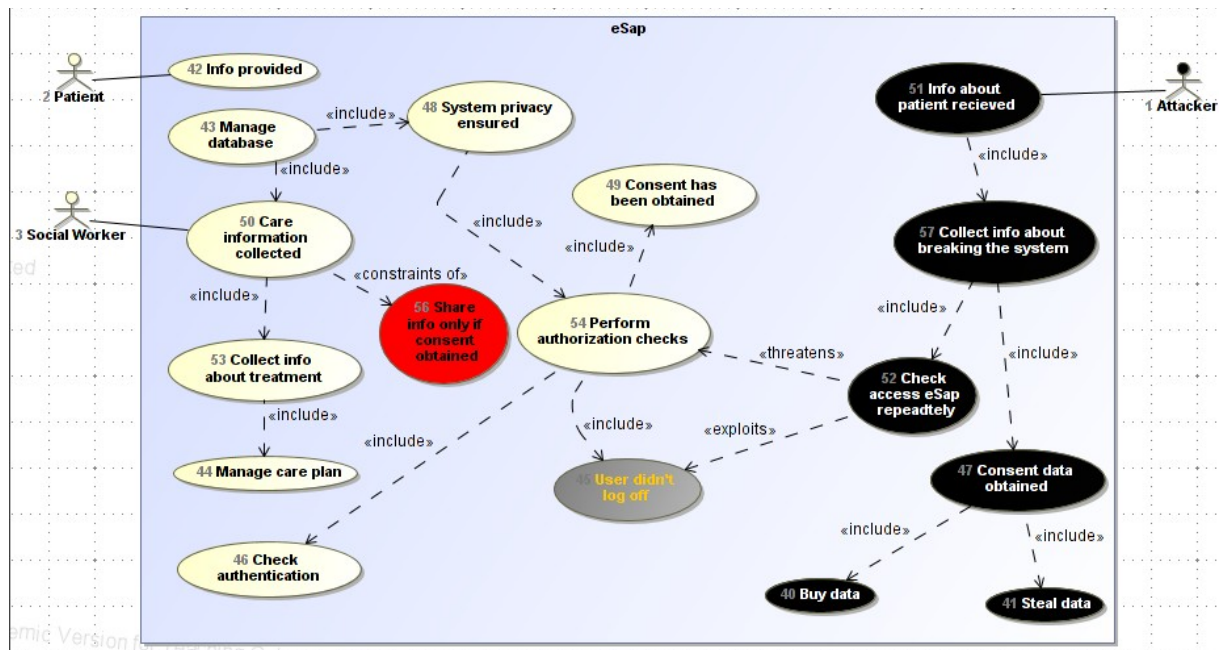


Figure 24: STMC8 rule example

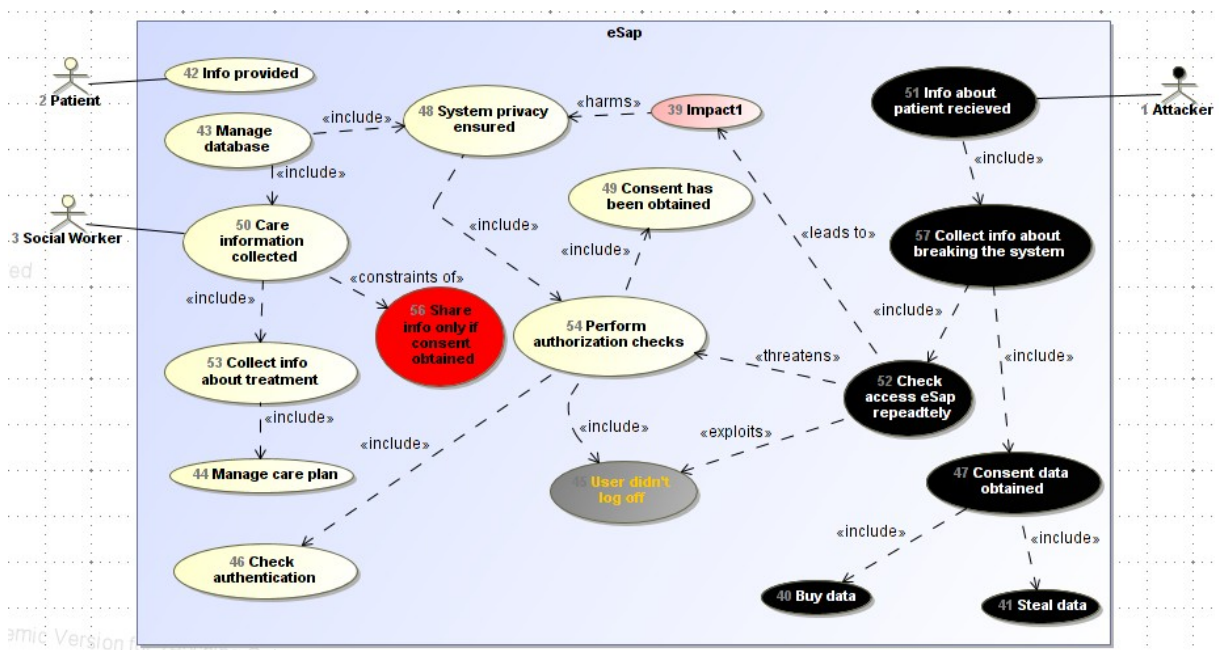


Figure 25: STMC9 rule example

Table 11: Examples of Misuse case constructs transformed to Secure Tropos constructs

Rule ID	Misuse case construct (before)	Secure Tropos construct (after)	Figure
Asset-related concepts			
MCST1	<i>System boundary</i>	System actor	Figure 26
MCST2	<i>Use case</i>	Goal	Figure 27
MCST2	<i>Use case</i>	Plan	Figure 27
MCST2	<i>Include relationship</i>	<i>Decomposition and means-end relations</i>	Figure 27
MCST3	<i>Actor</i>	<i>Actor</i>	Figure 28
MCST4	Security criterion	Security constraint	Figure 29
MCST4	<i>Constraints of relationship</i>	<i>Restricts relationship</i>	Figure 29
MCST5	<i>Association relationship</i>	<i>Dependency relationship with security constraint and goal</i>	Figure 30
Risk-related concepts			
MCST6	<i>Threatens relationship</i>	<i>Threat element</i>	Figure 31
MCST7	<i>Misuser</i>	<i>Actor</i>	Figure 32
MCST8	<i>Misuse case</i>	<i>Goal</i>	Figure 33
MCST8	<i>Misuse case</i>	<i>Plan</i>	Figure 33
MCST8	<i>Misuse case</i>	<i>Attack method</i>	Figure 33
MCST8	<i>Include relationships</i>	<i>Means-end and decomposition relations</i>	Figure 33
MCST9	<i>Vulnerability</i>	<i>Vulnerability</i>	Figure 8
MCST9	<i>Exploits relationship</i>	<i>Attacks relationship</i>	Figure 8
MCST10	Construct of <i>leads to, harms</i> relationships and <i>impact</i> element	<i>Impacts relationship from threat towards goal or plan</i>	Figure 34
Risk treatment-related concepts			
MCST11	<i>Security use case</i>	<i>Security constraint</i>	Figure 9
MCST12	<i>Mitigates relationship</i>	<i>Mitigates relationship</i>	Figure 9

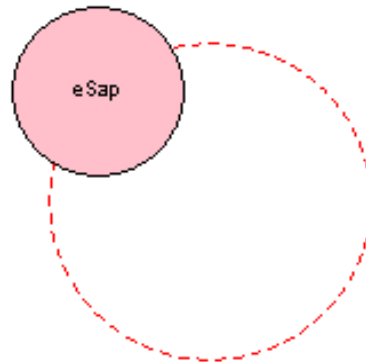


Figure 26: MCST1 rule example (Security Requirements view)

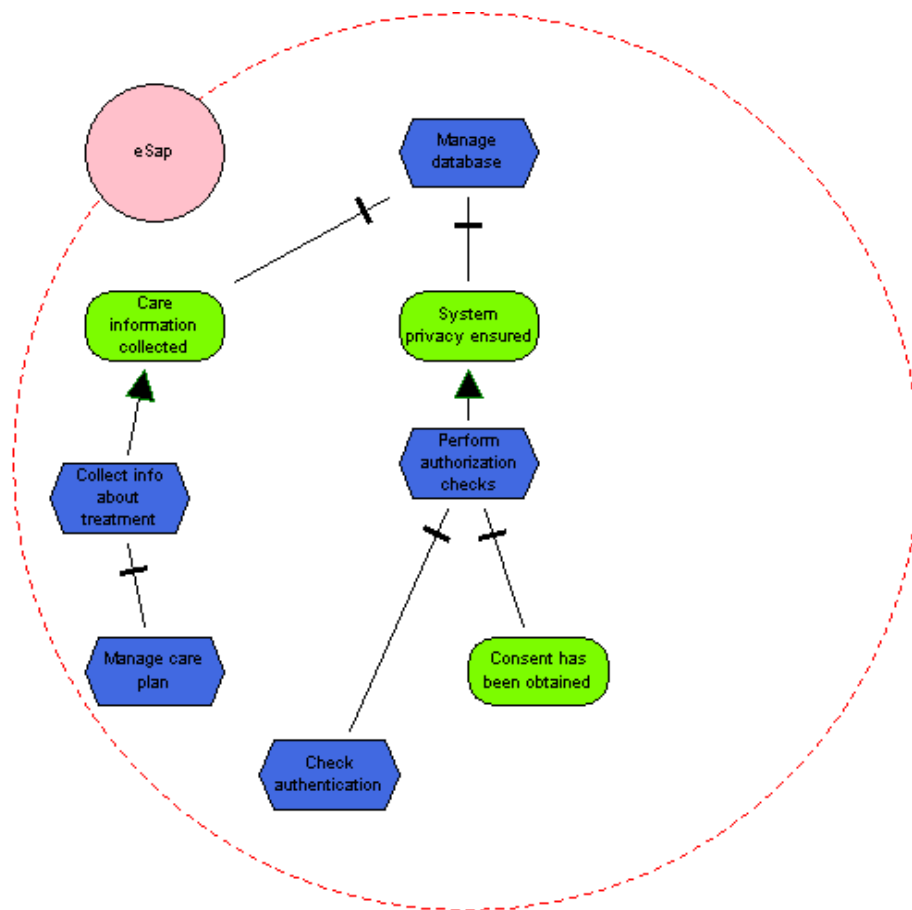


Figure 27: MCST2 rule example (Security Requirements view)

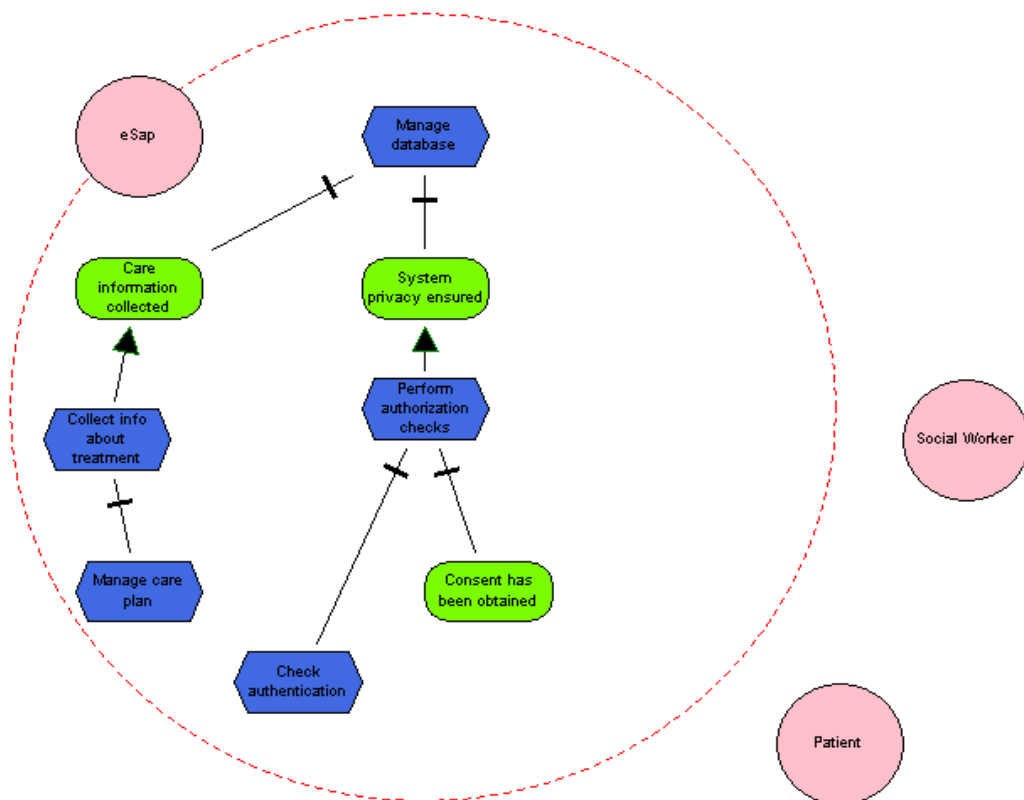


Figure 28: MCST3 rule example (Security Requirements view)

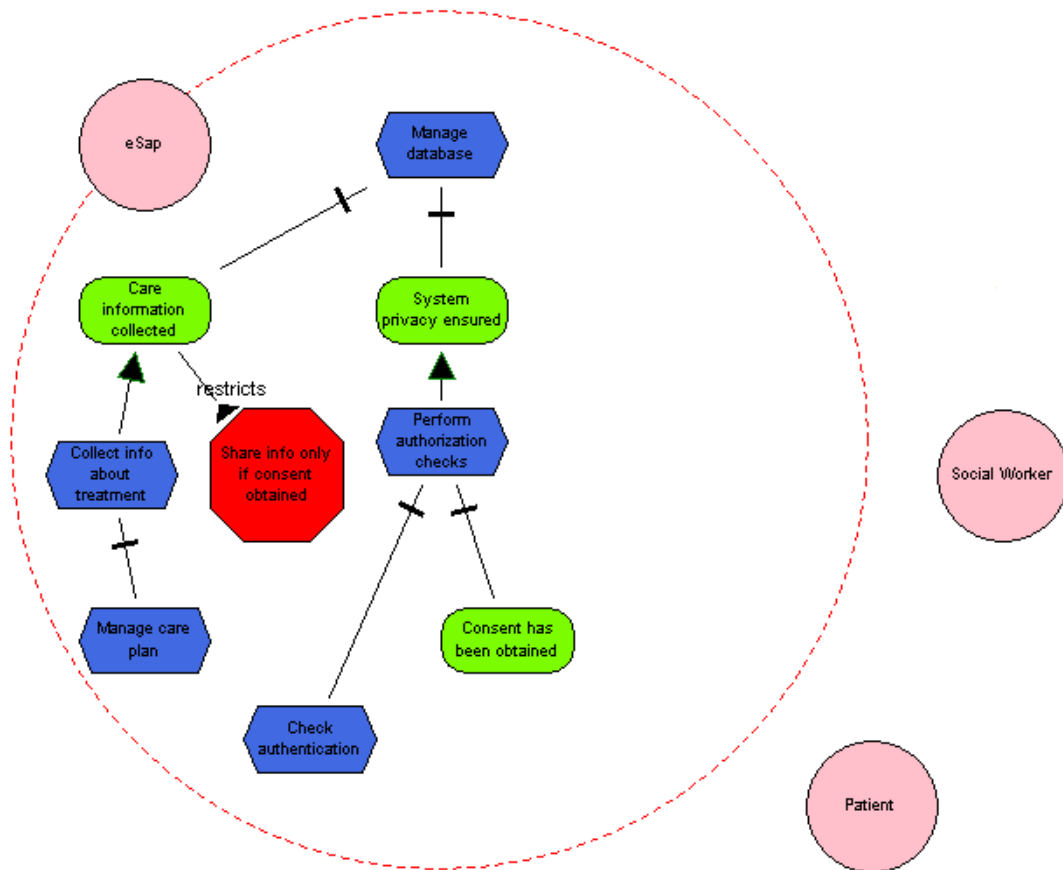


Figure 29: MCST4 rule example (Security Requirements view)

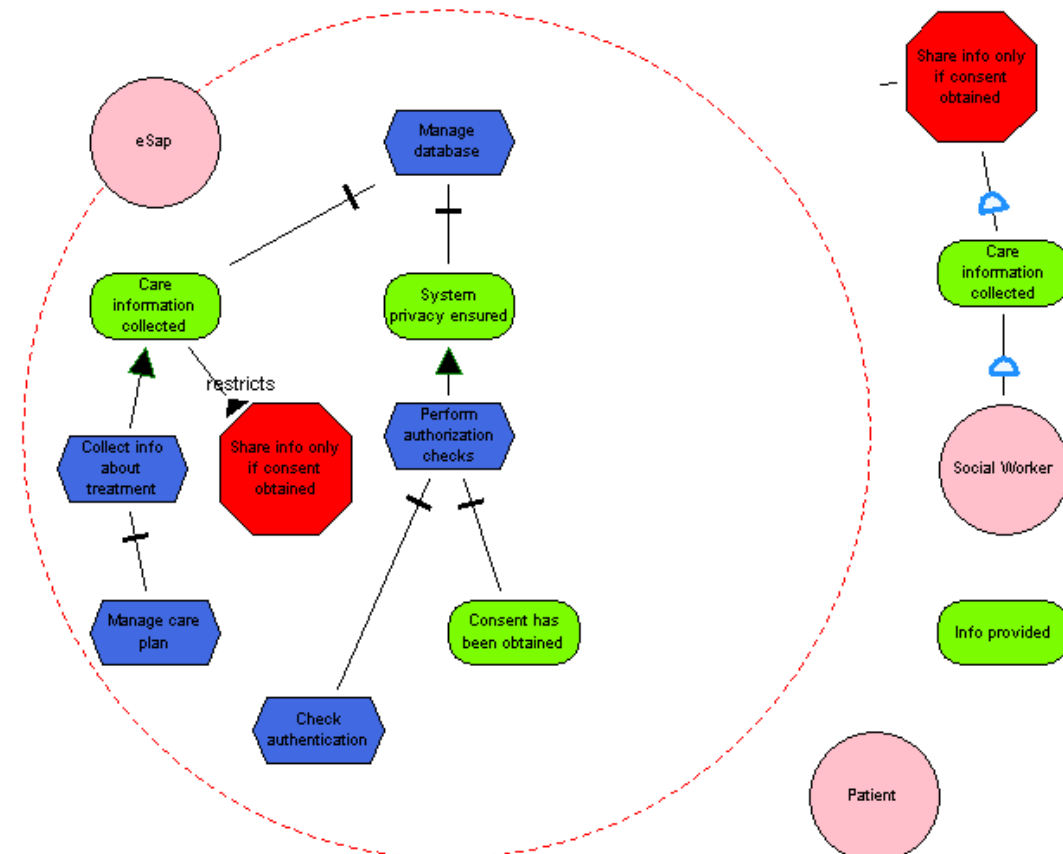


Figure 30: MCST5 rule example (Security Requirements view)

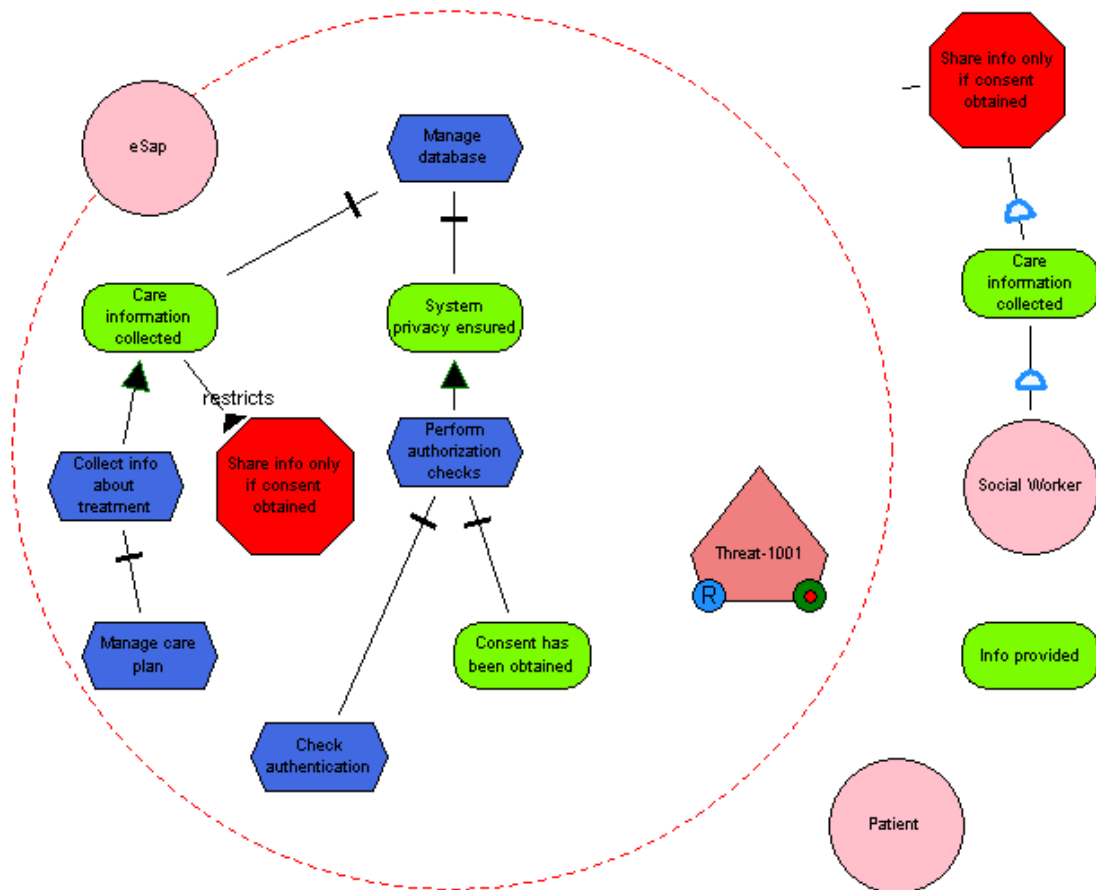


Figure 31: MCST6 rule example (Security Requirements view)

SRV - "Threat-1001"

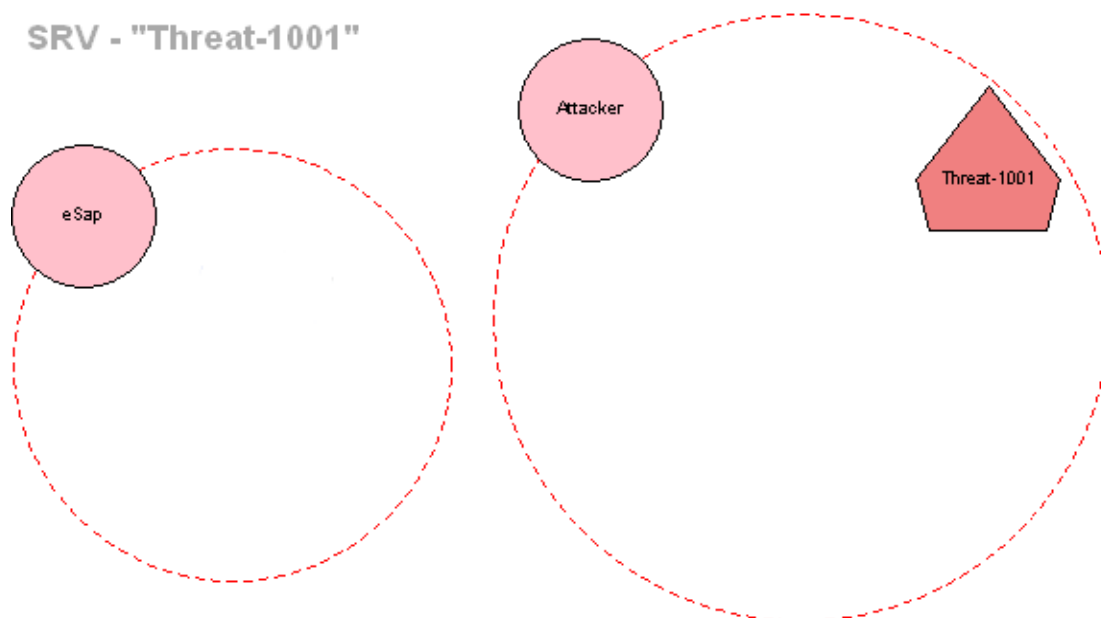


Figure 32: MCST7 rule example (Threat "Threat-1001" sub-view)

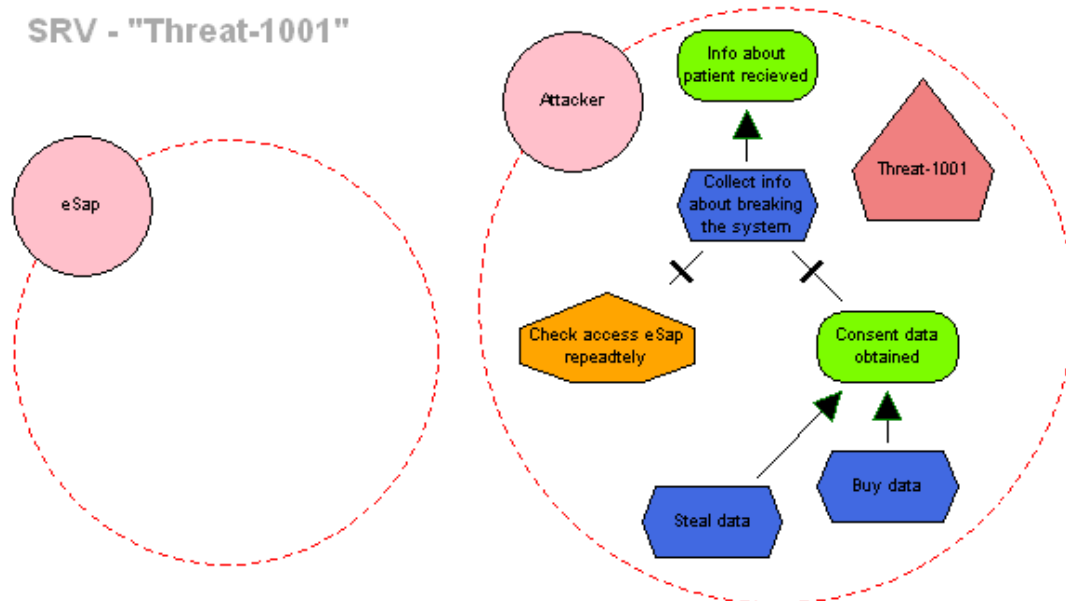


Figure 33: MCST8 rule example (Threat "Threat-1001" sub-view)

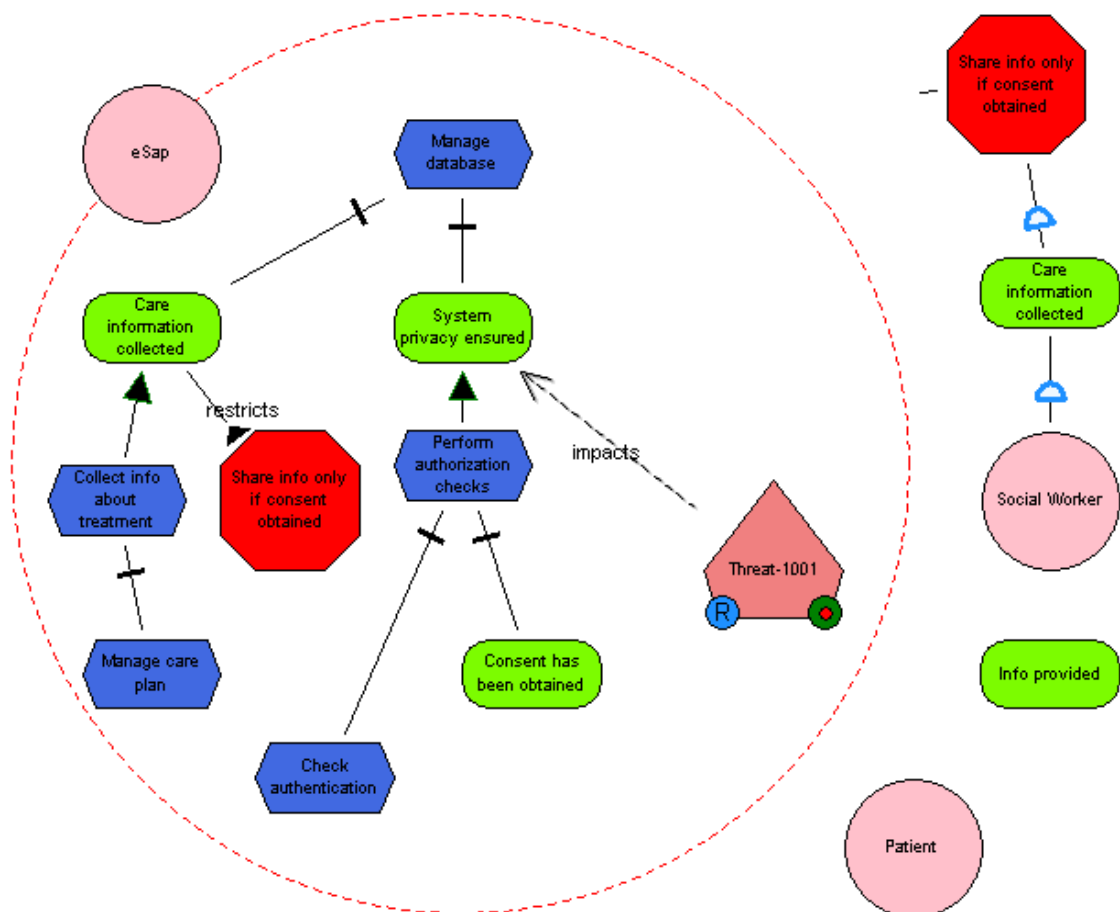


Figure 34: MCST10 rule example (Security Requirements view)

Appendix C: Validation Questionnaire

In your subjective opinion, please rate the following statements in 1-3 scale, comparing manual transformation process over automatic (place *X* in suitable table cell), where:

- (1) **Partially** (Easier/Efficient/Preferable)
- (2) **Slightly** (Easier/Efficient/Preferable)
- (3) **Fairly** (Easier/Efficient/Preferable)

NOTE! There can be only one “X” for one row (ST to MUC; MUC to ST) (ST – Secure Tropos; MUC – Misuse cases)

1. Which transformation process was easier to understand?						
	Manual			Automatic		
	3	2	1	1	2	3
ST to MUC						
MUC to ST						

2. Which transformation process is easier to learn?						
	Manual			Automatic		
	3	2	1	1	2	3
ST to MUC						
MUC to ST						

3. Which transformation process is more efficient(faster) to use?						
	Manual			Automatic		
	3	2	1	1	2	3
ST to MUC						
MUC to ST						

4. Which transformation process was/is easier to remember?						
	Manual			Automatic		
	3	2	1	1	2	3
ST to MUC						
MUC to ST						

5. Which transformation process You would prefer to use in future?						
	Manual			Automatic		
	3	2	1	1	2	3
ST to MUC						
MUC to ST						

Appendix D: Models for Validation

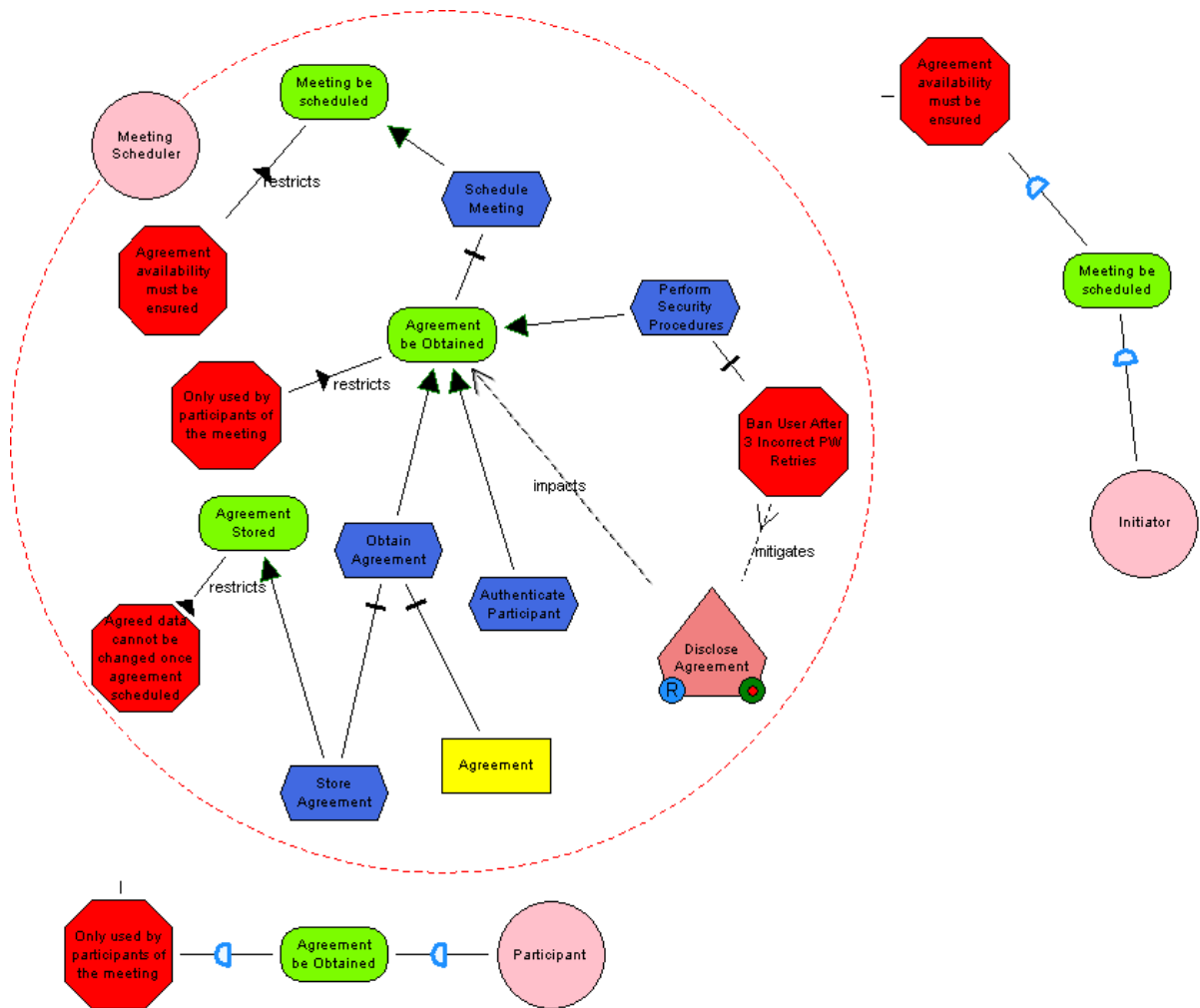


Figure 35: „Meeting scheduler“ Secure Tropos model example for validation

SRV - "Disclose Agreement"

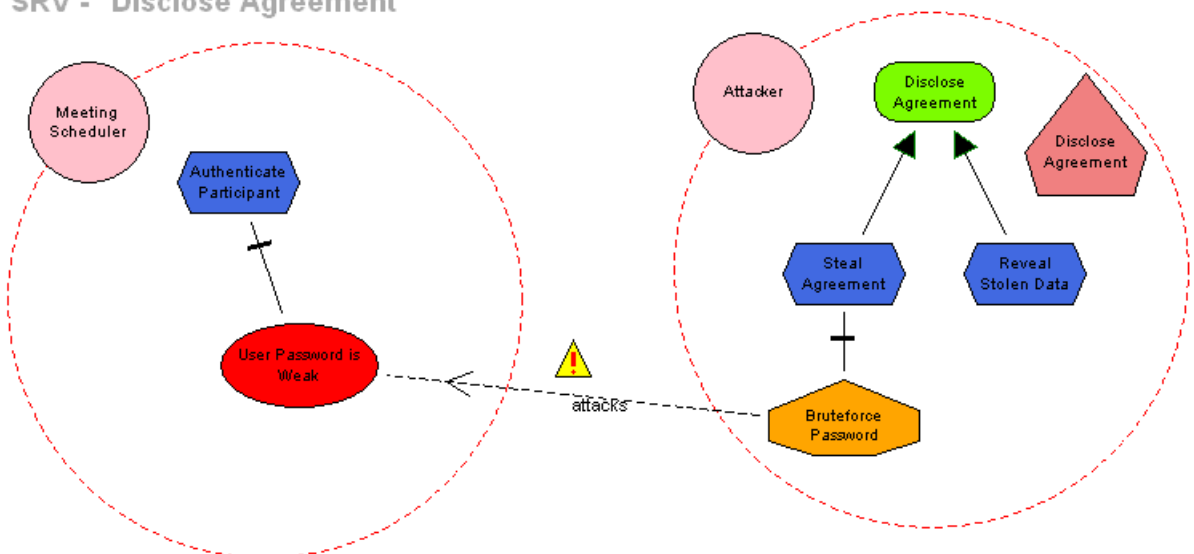


Figure 36: "Disclose Agreement" Threat sub-view

Non-exclusive licence to reproduce thesis and make thesis public

I, **Eduard Sing** (date of birth: 04.03.1993),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
 - 1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
 - 1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

A Prototype to Transform Models of Secure Tropos and Misuse Case Diagrams,

supervised by **Dr. Raimundas Matulevičius**,

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2014